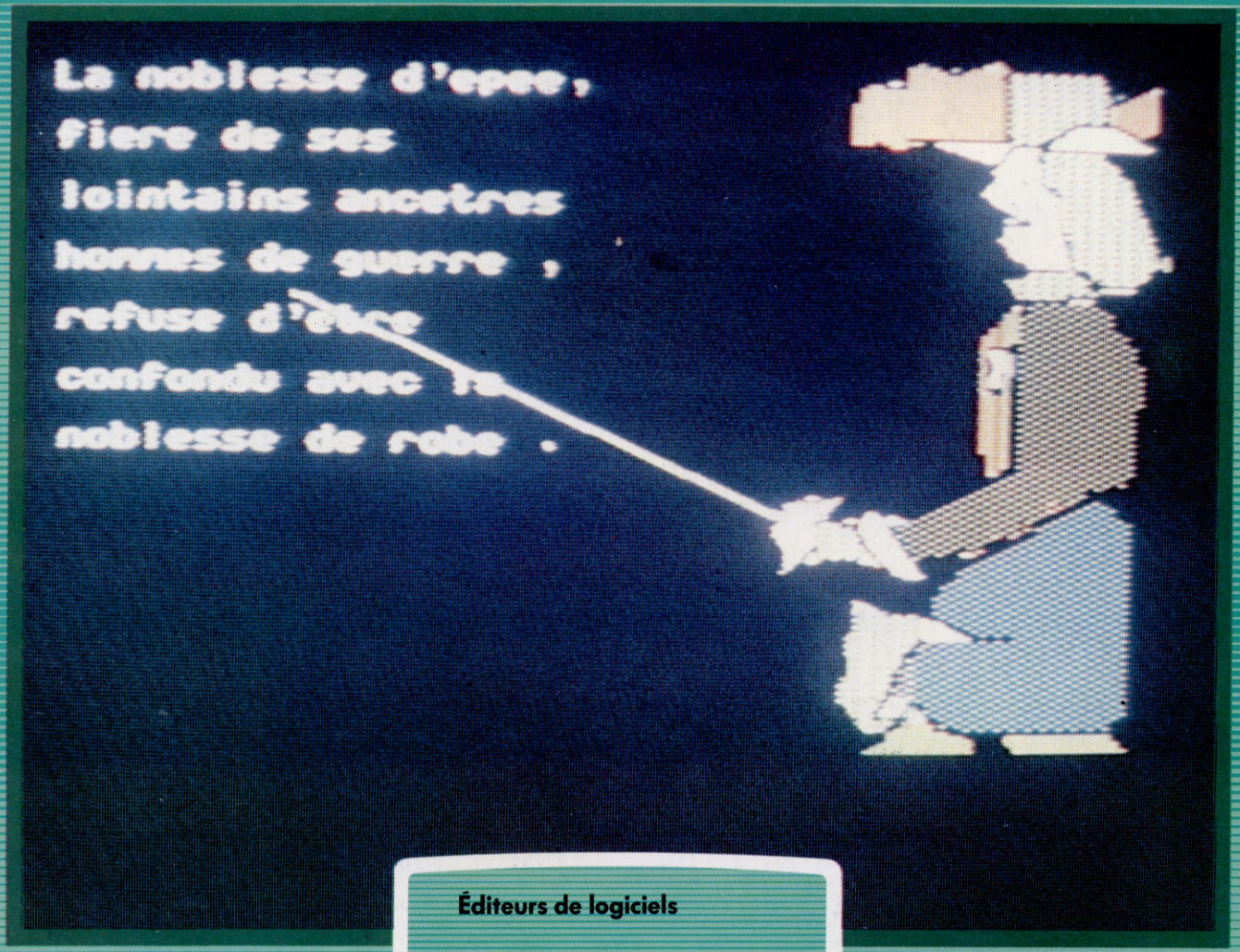
ABC INFORMATIQUE

N° 44

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Éditeurs de logiciels

Les algorithmes

L'Adam de Coleco

Créer une fenêtre sur écran

EDITIONS ATLAS

M6062-44-12F 85FB-3,80FS-$1.95

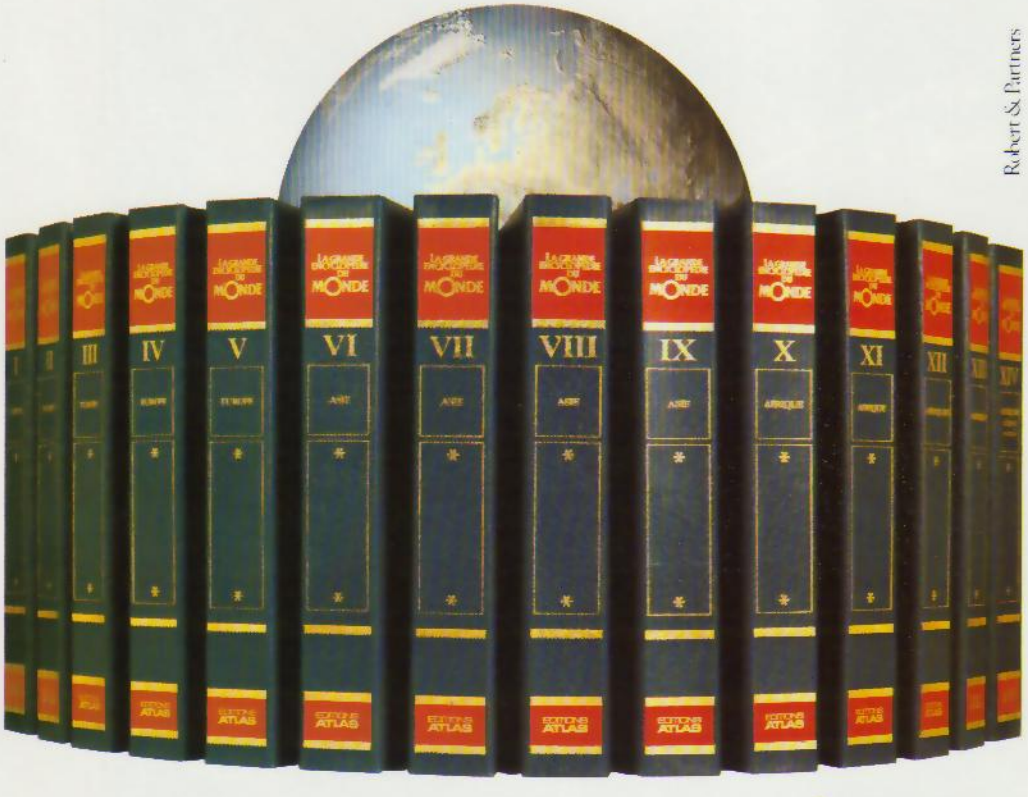
LA GRANDE ENCYCLOPEDIE DU MONDE

Pour la première fois dans votre bibliothèque, un fabuleux outil de connaissance et d'exploration.

14 volumes, plus de 15 000 illustrations couleur, des milliers de cartes et de données statistiques... Le Monde est là, tout entier dans votre main!

Au fil des pages, vous parcourez, continent par continent, pays par pays, toutes les contrées du globe, des rivages les plus hostiles aux contrées les plus accueillantes, vous enrichissez votre bibliothèque des trésors et des coutumes du monde entier.

Vous serez heureux de pouvoir consulter à tout moment "La Grande Encyclopédie du Monde", d'y retrouver un détail économique, un nom oublié, un lieu perdu, d'y vérifier vos connaissances.



Robert & Partners

EDITIONS ATLAS

LA GRANDE ENCYCLOPEDIE DU MONDE

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 28-10-10. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS, Z.I. de Lucé, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 28-10-10.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume IV, n° 44.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Claire Bischoff *(correction)*.
Crédit photographique, couverture : Jean Riby-Éd. Atlas.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : novembre 1984. 168411. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Écrire pour l'écran

Des éditeurs de logiciels disposent de moyens importants pour rechercher le succès. Certains possèdent des équipements de plusieurs millions de francs pour mettre au point leurs produits.

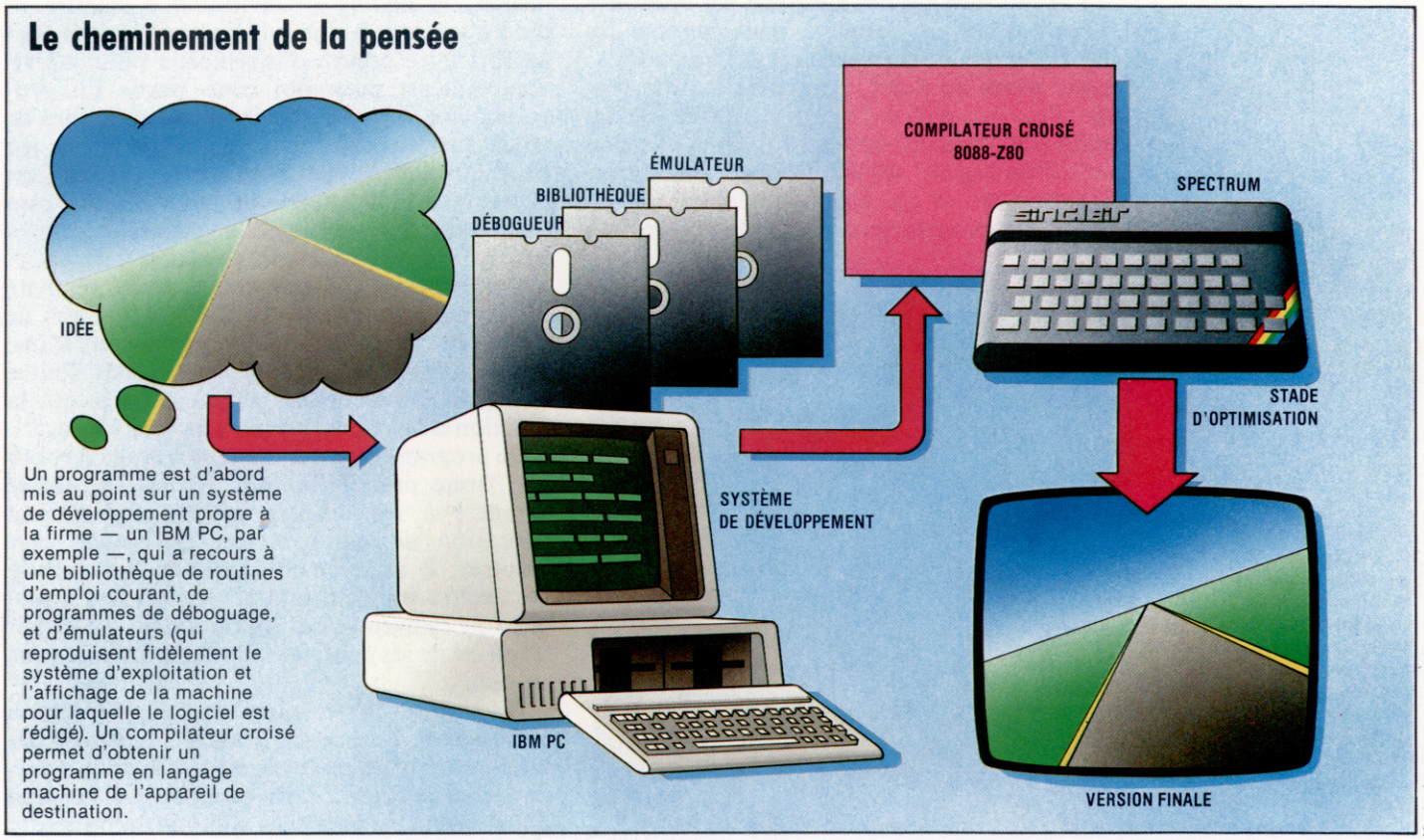
Bien entendu, un équipement coûteux ne suffit pas à garantir le succès d'un logiciel; des amateurs ont bel et bien réussi à gagner gros avec des programmes écrits chez eux sur un simple ordinateur familial. Pourtant, les Mozart de l'informatique seront bientôt une espèce en voie de disparition, vu le développement, ces dernières années, de grosses compagnies. Elles sont armées d'ordinateurs puissants et d'utilitaires sophistiqués, ce qui leur donne un net avantage et accroît la productivité de leurs programmeurs.

Tout logiciel destiné aux micro-ordinateurs personnels doit faire preuve d'une certaine vitesse d'exécution; ce qui signifie qu'il doit, au moins en partie, être rédigé en langage machine. Mais ce dernier est très peu maniable, et il faut utiliser d'autres programmes pour en tirer parti. Au minimum, on aura besoin d'un assembleur pour traduire le programme source en programme objet, le seul que la machine puisse comprendre, et c'est une tâche redoutable si le logiciel est de grandes dimensions. Beaucoup de programmeurs travaillent de cette façon; pour écrire un programme destiné à un ordinateur donné, ils ont recours à un assembleur spécifique à cet appareil. Mais cette méthode a des limites étroites.

La première est la faible qualité des programmes assembleur disponibles pour les micro-ordinateurs. Même le plus simple d'entre eux consomme de grandes quantités d'espace mémoire, et limite par conséquent la taille des programmes que l'on peut créer. Par ailleurs, il est difficile de travailler longtemps sur bien des machines existantes : clavier et affichage médiocres, et parfois absence de lecteurs de disquettes, ce qui rend très compliqué leur usage prolongé.

C'est bien pourquoi la plupart des firmes professionnelles procèdent autrement : elles font usage d'ordinateurs de gestion pourvus de logiciels appropriés (qu'on appelle « systèmes de développement »). Les programmeurs qui en font usage écrivent souvent en PASCAL ou en langage C, dans des versions dites « compilateurs (ou assembleurs) croisés ». Cela veut dire

Le cheminement de la pensée



Un programme est d'abord mis au point sur un système de développement propre à la firme — un IBM PC, par exemple —, qui a recours à une bibliothèque de routines d'emploi courant, de programmes de déboguage, et d'émulateurs (qui reproduisent fidèlement le système d'exploitation et l'affichage de la machine pour laquelle le logiciel est rédigé). Un compilateur croisé permet d'obtenir un programme en langage machine de l'appareil de destination.

Kevin Jones

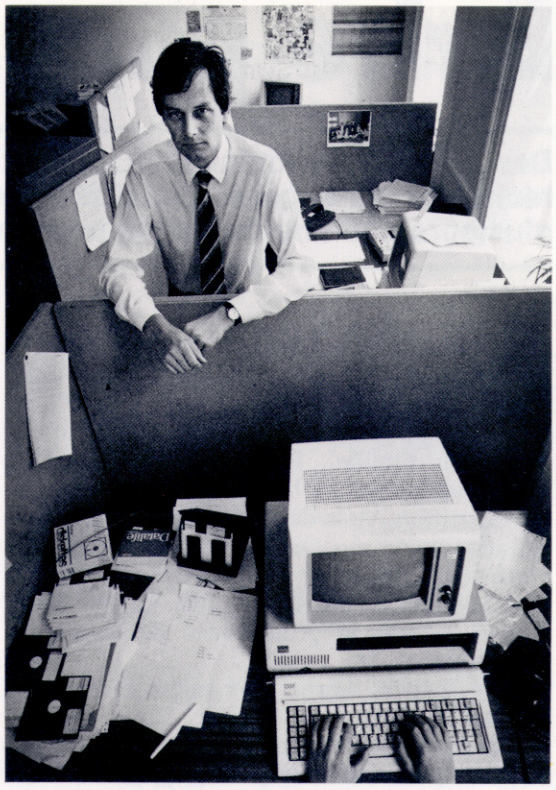
Intelligent Software
Spécialisée dans les jeux de stratégie tels que les échecs, la compagnie fait usage de IBM PC et de Apple, grâce à des interfaces spécialement construites.
La segmentation des programmes permet à IS de travailler plus commodément pour de nombreux ordinateurs, ainsi que pour les jeux d'échecs électroniques proprement dits.
(Cl. Tony Sleep.)

que le travail est effectué sur une machine construite autour d'un microprocesseur 8086, par exemple, alors que le programme à rédiger tournera sur un Z80. Ces compilateurs croisés sont des langages évolués (comme le BASIC), ce qui rend leur emploi facile; mais ils génèrent directement du langage machine.

Il est évident qu'un système de développement représente un pas en avant énorme par rapport à un simple micro-ordinateur. Un assembleur installé sur disquette, ou qui utilise un espace RAM étendu pour avoir plus de place, est bien plus efficace que son équivalent sur cassette, qui opère au sein d'un appareil plus limité. On peut même lui adjoindre des routines de déboguage sans se préoccuper de dépasser les limites de la mémoire. Un ordinateur de gestion a par ailleurs un bon clavier, un affichage excellent et des lecteurs de disquettes plus élaborés.

Un bon exemple d'emploi de ce type de méthodes est donné par Intelligent Software (IS), une firme fondée en 1981 en Grande-Bretagne et qui met en commun l'expérience de David Levy, un grand joueur d'échecs, et de Robert Madge et sa compagnie ANT Microware. Elle se spécialise dans les jeux de stratégie, travaillant au contrat pour la plupart des micro-ordinateurs les plus répandus. Elle se charge aussi de rédiger des programmes performants pour les jeux d'échecs électroniques. Ceux-ci, tout comme les logiciels de bridge, n'ont guère besoin d'un affichage perpétuellement modifié; mais il leur faut, par contre, énormément de calculs. C'est pour cette raison qu'ils leur faut les services d'un assembleur, tout comme pour les jeux d'arcades.

IS se sert bien sûr des machines sur lesquelles tourneront ses programmes, mais emploie des IBM PC et des Apple munis d'interfaces spécialement mises au point, pour permettre le passage d'un appareil à l'autre. En effet, on lui confie fréquemment des travaux de conversion (ainsi d'un logiciel de jeu d'échecs prévu pour tel ordinateur, et qui doit être adapté pour un autre). Ses programmeurs ont donc appris à



rédiger des programmes qu'on peut segmenter aisément. Une technique utile dans ce cas, lorsque le temps presse, est de diviser le programme en deux parties : le jeu et son déroulement d'une part, les entrées et sorties de l'autre. Ces dernières varient en effet d'un ordinateur à l'autre, les adresses mémoire et les ports ne seront pas les mêmes, le *polling* (« invitation à émettre ») cédera la place à une interruption, etc. Il faut parfois faire preuve d'ingéniosité pour y arriver, et c'est pourquoi cette partie du programme ne doit pas excéder certaines limites de taille. En revanche, tout ce qui a trait au jeu peut être développé largement, puisque cet aspect reste indépendant du matériel (sauf, bien sûr, du microprocesseur); la conversion pourra se faire sans difficultés importantes.

Comme IS ne veut pas restreindre la créativité de ses programmeurs, elle a établi très peu de règles pour la rédaction des programmes. L'une d'elles, cependant, précise que le code source doit être abondamment commenté, afin que la fonction de chaque routine soit bien claire.

Le programmeur qui travaille à domicile pour une firme productrice de logiciels, et qui se consacre à ses propres projets, a rarement l'occasion de partager son expérience avec d'autres. Il préserve son individualité au prix d'un effort supplémentaire, car il lui faut à chaque fois réinventer des routines déjà créées par tel ou tel de ses confrères, dans un but analogue au sien.

Une autre firme anglaise, Psion (comme, entre autres, Loriciel en France) a recourt à des mini-ordinateurs pour mener à bien ses travaux de développement. Elle possède un matériel d'une valeur de plusieurs millions de francs.

Visions
Cette firme a recours essentiellement à des programmeurs qui travaillent à domicile, sur les ordinateurs de destination eux-mêmes. Une fois l'idée de base et le scénario mis au point, les routines sont élaborées dans le langage assembleur correspondant (Z80 ou 6502).
(Cl. Bob Bromide.)

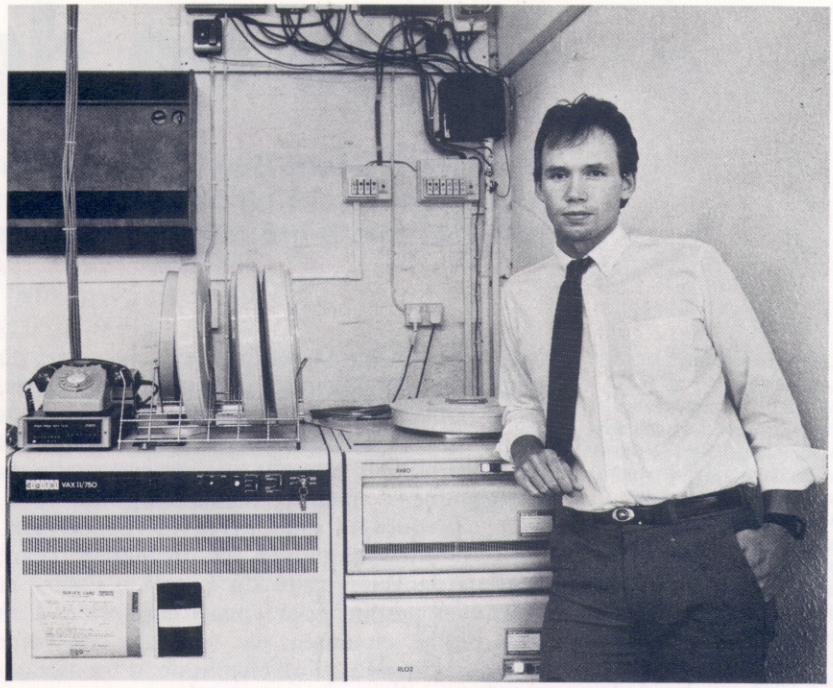
Elle a pourtant commencé par produire des logiciels destinés au ZX81, en utilisant cet appareil lui-même ! Lorsqu'elle fut chargée de créer la cassette accompagnant chaque Spectrum, elle fit l'acquisition d'un TRS-80 (qui a le même microprocesseur Z80) équipé de lecteurs de disquettes, et construisit une interface spéciale reliant les deux machines. En août 1982, toutefois, la compagnie décida de ne plus élaborer un nouveau système de développement à chaque fois qu'un micro-ordinateur faisait son apparition sur le marché, et consacra une part importante de ses profits à l'achat de gros matériel utilisable en temps partagé, qui pourrait, au moins théoriquement, s'adapter à tous les modèles qui seront commercialisés dans l'avenir. Elle porta son choix sur deux Vax 750, dotés du système d'exploitation VMS de DEC (Digital Equipment Corporation).

Ils offraient deux grands avantages : la qualité des logiciels fournis par DEC, qui permettent la création d'aides à la programmation spécifiques, et la très grande puissance du matériel et du système d'exploitation. Psion pouvait ainsi rassembler une vaste collection de programmes particuliers : compilateurs, bibliothèques de sous-routines employées fréquemment, utilitaires de recherche d'erreurs, tout cela étant accessible simultanément aux quinze ou vingt programmeurs qui travaillent en même temps sur la même machine. Les deux Vax peuvent d'ailleurs aisément échanger leurs informations quand c'est nécessaire.

Psion avait déjà recouru à une idée de ce genre, du temps où elle faisait usage d'un TRS-80, mais l'échange des données entre les disquettes était très vite devenu lourd et fastidieux à partir d'un double lecteur de disquettes. Les deux nouveaux appareils suppriment ces contraintes ; on peut travailler à plusieurs, appeler les modules dont on a besoin en un temps minimum ; il n'est même pas besoin de travailler au même projet. C'est là le gros avantage du système en temps partagé. Psion envisage même l'achat d'un troisième Vax, qui prendrait en charge les tâches administratives et qui permettrait l'affectation des deux autres à la seule programmation.

Vous ruer immédiatement pour acquérir un Vax — à supposer que vous en ayiez les moyens — ne suffirait pourtant pas à vous mettre sur un pied d'égalité avec Psion. Car la firme ne s'est pas limitée à l'emploi des logiciels fournis par DEC ; un très gros travail s'est révélé nécessaire pour mettre au point des routines simples, mais efficaces et fiables, et de nombreux utilitaires et aides à la programmation, généralement rédigés en langage C.

Psion affectionne ce dernier langage, qui peut produire des programmes objets assez compacts et rapides pour des microprocesseurs 16 bits comme le 8086 ; mais c'est loin d'être le cas pour les 8 bits. Comme la compagnie doit travailler à destination du Spectrum, il lui a fallu élaborer ses propres techniques, et bien qu'elle soit avare de confidences, on sait que le compilateur qu'elle utilise a été écrit en C ; il est aisément transférable entre machines différentes, et génère un code extrêmement efficace.



Psion
La compagnie a fait l'achat en 1982 de deux mini-ordinateurs Vax 750, qui constituent la base de développement des programmes qu'elle crée. Chaque machine peut accueillir jusqu'à 20 programmeurs à la fois, et leur permet l'accès à des compilateurs croisés, à des bibliothèques de routines, à des logiciels de déboguage, pour créer et pour convertir des programmes.
(Cl. Tony Sleep.)

C'est une règle universelle qui veut que trente pour cent de tout le travail de programmation soit consacré à la maintenance des systèmes et à la création d'aides à la programmation spécifiques, mais Psion pense que l'effort en vaut la peine. Rédiger soi-même le programme source, c'est en être totalement propriétaire.

Psion a pourtant fait l'acquisition de simulateurs, qui reproduisent exactement le fonctionnement de microprocesseurs comme le 6502 ou le Z80. Les gros ordinateurs Vax peuvent ainsi se comporter comme des Commodore 64 ou des Spectrum, mais il est vrai qu'en dépit de toute leur puissance, le logiciel tournera bien plus lentement que la machine qu'il imite. Le programmeur peut cependant voir exactement quel est le contenu de chaque adresse mémoire, à n'importe quel moment, sans attendre l'adaptation définitive : c'est particulièrement utile en cas d'erreurs de programmations. Ordinairement, lorsqu'un programme en langage machine se plante, il est difficile d'en trouver la cause exacte. Psion s'épargne ainsi de longues heures de déboguage.

Elle a consacré récemment beaucoup de travail à la mise au point des quatre utilitaires qui accompagnent le QL. Celui-ci est équipé d'un microprocesseur de la famille du 68000 de Motorola, conçu en fonction des langages évolués, et les programmes écrits en C font avec lui un si remarquable travail de compilation que le recours à l'assembleur devient inutile. Si tous les nouveaux ordinateurs suivaient l'exemple du QL, le langage C pourrait même devenir le seul outil indispensable, et Psion, tout comme les autres petites compagnies, pourrait enfin quitter cet univers à la Dickens où l'on travaille encore « à la main ».

Drôle de donjon

Avec MUD (Multi User Dungeon/Donjon Multi-Utilisateur), plusieurs joueurs pourront se connecter à un ordinateur central depuis leur terminal micro-ordinateur, et entreprendre une partie.

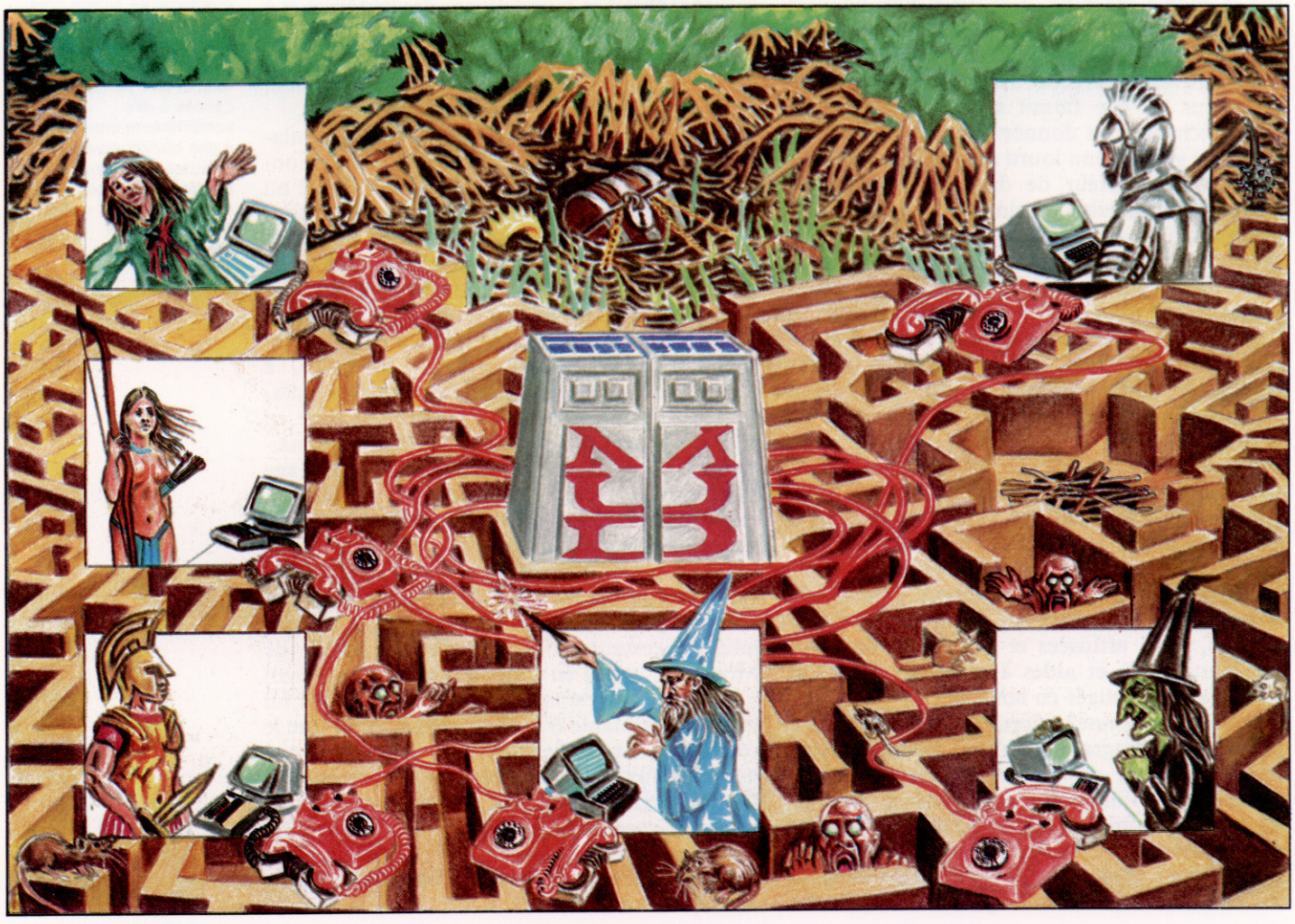
Lutte sans merci
Les jeux sur ordinateurs centraux permettent l'engagement de nombreux participants. Dans le cas de MUD, ils peuvent atteindre le nombre 43. Guerriers, enchanteurs, et autres personnages peuvent se combattre ou s'allier pour conquérir le trésor et devenir des magiciens ou des sorciers tout-puissants (!). Les jeux d'aventures sur ordinateur central impliquent de nombreux scénarios détaillés. Leurs théâtres d'opérations sont variés. Partout, il peut y avoir des trésors et des farfadets. (Cl. Adrian Morgan.)

MUD est un jeu d'aventures en temps réel où vous faites connaissance avec d'autres joueurs. Vous pouvez leur demander des conseils, vous liguer contre un adversaire commun, ou les attaquer bille en tête. Ils ne font pas partie du programme et jouent en même temps que vous. C'est pourquoi leurs initiatives vous concernent.

MUD fonctionne sur un ordinateur géant DEC10 de l'université du comté d'Essex, et vous avez besoin, pour vous y connecter, d'un programme émulateur de terminal sur votre micro-ordinateur, d'un téléphone, d'un modem et d'un système de commutation par paquets qui comptabilise vos accès (Packet Switching System). Un programme émulateur de terminal permet à votre ordinateur personnel de communiquer avec l'ordinateur central par liaison téléphonique. Vous pouvez soit écrire votre propre programme d'émulation, soit en acquérir un. De manière idéale, il doit permettre le défilement de l'écran ainsi qu'une longueur de ligne de 80 caractères. Le logiciel Micronet concerne l'Angleterre et serait comparable en France au Vidéotext dans la mesure même ou l'affichage est, dans les deux cas, particulier.

Il est préférable d'utiliser des progiciels de communication, Terni et Communicator par exemple, qui figurent sur des composants de mémoire morte que l'on peut placer sur le BBC Micro.

Le modem doit être en conformité avec les normes de télécommunication en vigueur dans le pays. Si une telle expérience devait être adaptée pour la France, les cadences en bauds



devraient être modifiées. En Angleterre, le système de commutation par paquets des Telecom britanniques est soit 300/300, soit 1200/75, soit 200/1200 bauds. Il faut également adjoindre un coupleur acoustique. Le réseau de transmission par paquets vous permet d'entrer en contact avec des interlocuteurs lointains.

La procédure d'accès au réseau est directe : le logiciel d'émulation du terminal étant en exploitation, vous faites le numéro d'accès au réseau qui vous est attribué, vous connectez votre modem et indiquez votre « identité ». Il faut ensuite donner l'adresse pour le réseau de l'université où se trouve l'ordinateur central, le DEC10. Après s'être ainsi connecté, l'utilisateur lancera la commande RUN MUD.

Le programme est continuellement mis à jour, et l'utilisateur verra s'afficher en premier la date de la dernière version. Il devra ensuite choisir un nom de code, le nom de son personnage. Si c'est votre premier jeu, ce personnage sera créé et vous deviendrez pour le jeu un « novice ». Les niveaux sont les suivants :

Niveau	Points	Homme	Femme
1	0	novice	novice
2	400	guerrier	guerrière
3	800	héros	héroïne
4	1600	champion	championne
5	3200	super héros	super héroïne
6	6400	enchanteur	enchanteresse
7	12800	sorcier	sorcière
8	25600	nécromancier	nécromancière (!)
9	51200	légende	légende
10	102400	magicien	magicienne

Si vous parvenez à la fin d'un jeu sans être tué, les points que vous aurez gagnés seront ajoutés au programme, et vous pourrez continuer le jeu une prochaine fois sur cette base.

Si c'est vous qui êtes tué par un autre joueur, votre personnage est supprimé du jeu et vous devrez reprendre le combat en tant que « novice », sans aucun point (bien sûr). Aussi est-il toujours prudent d'y regarder à deux fois avant d'engager un combat.

Vous commencez le jeu sur le théâtre d'opération dit « Une route étroite entre des terres ». Si vous frappez "QUI EST LÀ?", la liste des personnes jouant en même temps que vous va s'afficher. Vous pouvez choisir de saluer l'un d'eux, en lui adressant un message du genre « Salut, Max, je suis nouveau et j'aurais besoin d'un coup de main ». Le message est alors transmis au terminal du joueur Max. Vous pouvez aussi vous adresser à tous les joueurs à la fois en tapant : « Attention je suis là, tas d'enflures, garez-vous », mais cela n'est pas vraiment indiqué de commencer ainsi lorsque l'on est novice...

Si vous tapez « A l'aide ! », vous obtiendrez quelques informations sur les mouvements et sur les commandes possibles. Les différents déplacements sont décrits de la sorte : « La plupart des commandes simples de déplacement sont admises, plus certaines que vous aurez à découvrir ! »

Les commandes disponibles peuvent être listées en tapant "COMMANDES". Cela représente trop d'informations défilant à l'écran pour pouvoir les prendre par écrit. Aussi, certains logiciels de terminal d'ordinateur vous rendront-ils possible la recopie de l'écran sur un disque à des fins de consultation ultérieure. Cela vous permettra également de pouvoir établir une carte appropriée des terres parcourues, carte que vous pourrez mettre à jour après chaque jeu. Voici les commandes possibles sur toute une journée de jeu :

COMMANDES (abréviations en majuscules)

AutoWho, < secondes > (Qui est là ?)	Back	BERSERK
BRIEF	BUG	BYE
CONVERSE	DRop < objet, du trésor >	DRop ALL
EMPTY < sac >...	EXITS	Flee < direction >
FOLLOW < nom >	Get < objet	Get ALL
REFUSE < nom >	du trésor >	
GIve < objet du trésor > TO < nom >		go < direction >
	HELP nom	HINTS
Help	HUG nom	INFO
HouRS	KEEP < objet	
Inventory	du trésor >	Kill < nom >
KISS < nom >	LEVEL	LOG
< level >; < message >	Look around	Look < sac >
Look < direction >	LOSE < nom >	MEDITATE
No PassWord	PassWord	PERSONNA
PronNouns	Quick Who	Quit
« < message > »	REFUSE < nom >	
RETaliate < item >	SAVE	SCore
SHout < message >	SLEEP	SPELL
STeal < objet du trésor > From < nom >		UNKEEP
tell < nom >; < message >		WHEN
VERBOSE	WEIGH < objet du trésor >	
WHO	WRITE < sujet >; < message >	

MUD est un jeu d'aventures très documenté, avec tous les détails et les descriptions nécessaires sur les emplacements. Lorsque l'utilisateur connaît bien le scénario, il peut taper BRIEF, et le programme lui épargnera les descriptions. Le graphisme du télétexte PRESTEL (en Angleterre seulement) est lent et peu précis (comme le Vidéotext en France). L'utilisateur préférera toujours un jeu fondé sur du texte à un jeu fondé sur des graphiques, bien que ce dernier soit une nouveauté intéressante. Un jeu faisant intervenir le texte seulement est beaucoup plus attrayant dans la mesure où il laisse une part plus grande à l'imagination. Un autre inconvénient d'un jeu purement graphique est qu'il suppose une version différente par modèle de micro-ordinateur, du fait des performances graphiques distinctes pour chaque machine.

MUD sera prochainement mis sur le marché en Angleterre. Il s'agit d'une expérience originale susceptible de développer l'utilisation du télé-traitement. Les auteurs de MUD, Richard Bartle et Roy Trubshaw, étudient une version pour ordinateurs VAX. Le jeu supposera une liaison téléphonique, mais pourra utiliser prochainement un câble si la demande est assez forte. En plus de l'acquisition du logiciel, l'utilisateur devra alors s'acquitter d'un abonnement au câble et sera taxé selon un tarif de consommation horaire dégressif.

Les maîtres
Pour plus de détails sur MUD, vous pouvez contacter : Richard Bartle, département des sciences de l'informatique, université d'Essex ; Colchester, Essex, Grande-Bretagne.

Éléments de base

Un algorithme est une suite d'instructions qui décrivent le déroulement d'un traitement informatique. En quoi la compréhension des principes de base des algorithmes aide-t-elle à la programmation?

Un algorithme définit un processus soit en termes de traitements déjà définis, soit en termes de traitements tellement élémentaires qu'ils n'ont pas besoin d'être définis. Ainsi, dans une recette, une instruction pourra être de préparer une sauce béchamel — la recette de cette dernière (son algorithme) ayant été donnée ailleurs dans le livre de cuisine. Une autre instruction pourra être « amener le mélange à ébullition », ce qui est supposé suffisamment explicite pour l'utilisateur de la recette. En termes de programmation, les algorithmes sont constitués d'instructions qui utilisent soit d'autres algorithmes (procédures, routines ou fonctions) écrits ailleurs dans le programme, soit des instructions du langage utilisé (commandes telles que PRINT et DIM ou fonctions mathématiques telles que LOG et TAN).

Cet article étudie la construction des algorithmes à partir d'autres algorithmes et éléments de base de traitement. Ces derniers sont les commandes et les fonctions du langage employé. A partir d'eux, on écrit de simples algorithmes destinés à accomplir des tâches simples (déplacer une figure graphique, ou prendre un nombre en entrée, par exemple).

Ces algorithmes servent à leur tour à élaborer des algorithmes plus généraux (mettre à jour l'affichage d'un jeu, commander un système de menu, etc.). Un autre niveau d'algorithme les intégrera en tant que parties d'un programme plus général.

Création d'algorithmes

Un algorithme a des entrées et des sorties. Cela signifie simplement que, comme traitement informatique, il accepte des données en entrée et produit des résultats. Les données initiales sont passées à l'algorithme sous la forme de paramètres, et proviennent de l'extérieur du programme. Les paramètres restent constants lors d'une même exploitation d'un algorithme. Affecter des paramètres est une démarche simple et familière, même à un programmeur débutant. Par exemple, le simple programme :

```
10 PRINT "Bonjour tout le monde!"
```

transmet le paramètre « Bonjour tout le monde! » à l'algorithme appelé par la commande PRINT. FNA(P), TAN(P), LEFT$(P$,5) et POKE P,5 sont d'autres exemples d'algorithmes avec P, P$ et 5 pour paramètres. De la même manière, les résultats d'un algorithme sont transmis en tant que paramètres. Lorsque le langage de programmation comporte des variables locales (comme le langage C et le PASCAL), les paramètres sont passés par un appel de procédure comme dans :

```
procédure(paramètre1, paramètre2, etc.);
```

La première étape consiste à définir les paramètres d'entrée et de sortie, c'est-à-dire leur contenu, leur type (valeur entière, nombre réel, chaîne de caractères, etc.), leur ordre de grandeur et les limites qui leur seront imposées.

Ensuite, il faut réfléchir à la manière de passer des entrées aux sorties, c'est-à-dire qu'il faut trouver le traitement le plus approprié. Il n'y a malheureusement pas de recette pour ces transformations qui nécessitent de la créativité et de l'ingéniosité. Il y a cependant des manières de faciliter ce processus.

Parmi celles-ci, la manière la plus évidente, et trop souvent négligée, est de copier un autre traitement ou, du moins, de s'en inspirer. Ainsi, au niveau le plus élémentaire d'algorithme, les fonctions intégrées du langage de programmation constituent autant de précieux algorithmes. Ce sont, par exemple, la gestion de chaînes de caractères, le traitement des fonctions trigonométriques, des entrées/sorties et, selon le langage, le processus de tri et de calcul matriciel. A part cela, l'algorithme recherché existe peut-être déjà dans un autre de vos programmes. Son code peut alors être intégré au nouveau. C'est pourquoi il est particulièrement important de créer votre propre bibliothèque d'algorithmes. Il existe en outre des publications d'algorithmes que l'on peut se procurer en librairie. Consultez à cet égard une bibliographie d'ouvrages de programmation en micro-informatique dans une librairie spécialisée.

Vous pouvez également « éplucher » les programmes publiés dans les revues de micro-informatique pour trouver des routines susceptibles d'être utilisables. Il existe enfin des algorithmes dont la destination première n'était pas informatique, et qui peuvent se révéler extrêmement profitables dans la mesure où leur domaine d'applications est compatible. Cette démarche de programmation s'applique à tous les domaines de spécialisation : ingénierie, électronique, mathématiques, etc.

Qu'il s'agisse d'adapter un algorithme existant ou d'en créer un au brouillon, il y a un certain nombre de critères que chaque instruction doit respecter. Une instruction doit être univoque et effective. Une instruction ne doit com-

porter aucune ambiguïté. Or, comme vous avez pu vous en apercevoir, il est très facile d'en introduire au départ, lorsque l'algorithme général est écrit sous forme d'un texte descriptif. Ainsi, les conjonctions « et » et « ou » n'ont pas la même extension dans la langue courante et en logique booléenne. Par exemple, si l'algorithme doit trouver tous les noms qui commencent par A *et* ceux qui commencent par B, vous pouvez penser qu'il suffit d'écrire :

```
IF PREMIÈRELETTRE="A" ET PREMIÈRELETTRE="B" THEN
```

Mais cela est faux parce qu'il est nécessaire dans ce cas de mettre un ET logique !

Une instruction doit toujours être suivie d'effet. Le programme ne doit pas comporter d'instructions impossibles. Une instruction est dite « effective » lorsqu'elle pourra être simulée sur le papier en un temps raisonnable. Cela signifie qu'une instruction du genre « soit X supérieur au plus grand nombre premier » n'est pas effective (il n'y a pas de plus grand nombre premier).

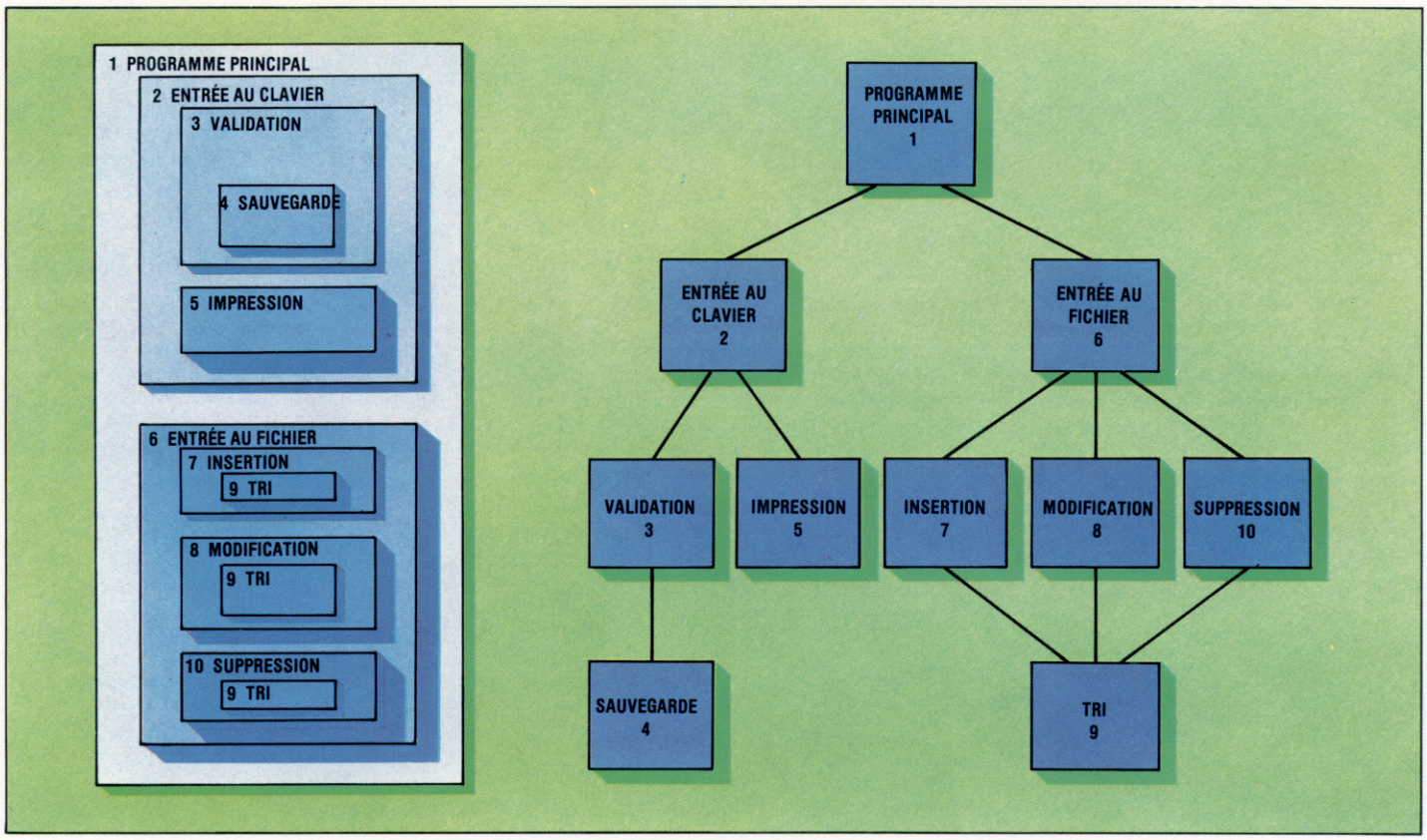
Il existe également des critères d'appréciation d'un algorithme dans sa totalité. Il doit comporter une fin. Celui que nous proposons ci-dessous en exemple n'en comporte pas et boucle sur lui-même :

```
step 1 let I equal 1
step 2 if I > 3 then exit
step 3 goto step 1
```

Il n'est pas toujours facile de savoir si un algorithme aura bien une fin. Cependant, les algorithmes qui comportent des boucles doivent satisfaire une condition pour s'achever (I>3 pour notre exemple). Il faudra s'assurer alors que cette condition est vérifiée (ce qui n'était pas le cas dans l'exemple donné).

Efficacité, généralité et élégance constituent autant de critères d'appréciation d'un algorithme. L'efficacité se mesure généralement en termes de temps de traitement et de place mémoire occupée. Ces deux aspects peuvent être compatibles. Ainsi, un code d'exécution rapide peut être également sobre en place mémoire. Sachez cependant que cela n'est pas toujours le cas. Une fois l'algorithme trouvé, il faut le rendre efficace. Il suffit parfois de modifier quelques détails. Un calcul sera plus rapide et occupera moins de mémoire en valeurs entières plutôt qu'en virgule flottante.

Le degré de généralité consiste pour l'algorithme à pouvoir traiter de nombreuses situations différentes de celle pour laquelle il était destiné. Il est utile à long terme de concevoir vos algorithmes comme devant être aussi généraux que possibles. Si un programme appelle plusieurs fois un message guide-opérateur sollicitant une réponse de type NON/OUI, il est avantageux de définir une seule et unique routine qui le fasse. Celle-ci devra demander à l'utilisateur : « Répondez par oui ou par non » ; elle devra lire la frappe au clavier, vérifier s'il s'agit de « oui » ou de « non » exclusivement ; dans le cas contraire, réitérer la question et afficher en dernier lieu la réponse appropriée. La routine peut avoir un degré de généralité encore plus grand si elle accepte pour différents programmes différents messages, selon les situations. L'élégance, pour un algorithme, signifie une solution à la fois simple et ingénieuse. Il est toujours préférable de rechercher l'efficacité avant l'élégance.



Pyramides et éléments de base
La structure du diagramme de gauche montre clairement la façon dont s'interpénètrent les algorithmes d'un programme, alors que le déroulement des procédures (diagramme de droite) met en relief les articulations et les niveaux de traitement de ce même programme. (Cl. Kevin Jones.)

Grand Prix

Laissez-vous aller dans l'un des plus grands succès des jeux d'arcades. Notez que ce jeu, écrit par Paul Bunn en basic pour Atari nécessite les manettes correspondantes.

Sur la piste, quatre automobiles s'affrontent dans une course folle; vous devez éviter l'accident tout en essayant de parcourir la plus grande distance. Utilisez le manche à balai pour diriger votre auto, et le bouton rouge pour changer de vitesse. Attention, ce programme utilise le langage machine, aussi sauvegardez-le avant de le lancer.

```
0 REM ** GRAND PRIX  ECRIT PAR  **
1 REM **        PAUL DUNNING     **
2 DIM S(4),V$(10),E$(100)
3 GRAPHICS 5:V$="LENT":POKE 752,1
5 GOSUB 1000
6 POKE 765,2:COLOR 2:PLOT 0,0:DRAWTO 79,
0:PLOT 79,47:DRAWTO 79,30:DRAWTO 0,30:PO
SITION 0,47:XIO 18,#6,0,0,"S:"
7 POKE 765,3:COLOR 3:PLOT 79,29:DRAWTO 7
9,1:DRAWTO 0,1:POSITION 0,29:XIO 18,#6,0
,0,"S:"
8 POKE 712,148
10 PP=PEEK(106)-16:POKE 54279,PP
20 PM=PP*256:POKE 559,62
21 FOR Q=53256 TO 53260:POKE Q,3:POKE Q-
8,RND(1)*255:NEXT Q
22 POKE 53248,100:POKE 53249,100:POKE 53
250,100:POKE 53251,50
23 POKE 704,109:POKE 705,89:POKE 706,79:
POKE 707,29:POKE 53277,3:POKE 559,62
24 FOR Q=PM+1024 TO PM+2048:POKE Q,0:NEX
T Q
30 RESTORE 210:FOR Q=PM+1144 TO PM+1280:
READ A:IF A=-1 THEN 32
31 POKE Q,A:NEXT Q
32 RESTORE 210:FOR Q=PM+1360 TO PM+1536:
READ A:IF A=-1 THEN 34
33 POKE Q,A:NEXT Q
34 RESTORE 210:FOR Q=PM+1576 TO PM+1792:
READ A:IF A=-1 THEN 36
35 POKE Q,A:NEXT Q
36 RESTORE 210:FOR Q=PM+1842 TO PM+2048:
READ A:IF A=-1 THEN 40
37 POKE Q,A:NEXT Q
40 POKE 623,1
50 SOUND 0,255,12,5:SOUND 1,145,12,5:SOU
ND 2,200,12,5
60 FOR Q=53256 TO 53260:POKE Q,3:POKE Q-
8,RND(1)*255:NEXT Q
70 POKE 704,109:POKE 705,89:POKE 706,79:
POKE 707,29:POKE 53277,3:POKE 559,62
75 ST=PM+1842:VI=INT(ST/256):LT=ST-256*V
I:POKE 203,LT:POKE 204,VI:POKE 205,10
80 P1=(RND(1)*100)+100:P2=(RND(1)*100)+1
00:P0=(RND(1)*100)+100:P3=50
85 FOR Q=0 TO 3:S(Q)=(RND(1)*2)+1:NEXT Q
87 POKE 53248,P0:POKE 53249,P1:POKE 5325
0,P2:POKE 53251,50
88 FOR Q=1 TO 50:X=USR(ADR(E$),7):NEXT Q
90 IF V$="VITE" THEN GOSUB 400:GOTO 120
92 POKE 53278,1
95 P0=P0+S(0):IF P0>255 THEN P0=0
98 X=USR(ADR(E$),STICK(0))
100 P1=P1+S(1):IF P1>255 THEN P1=0
105 X=USR(ADR(E$),STICK(0))
110 P2=P2+S(2):IF P2>255 THEN P2=0
120 KM=KM+0.01
124 X=USR(ADR(E$),STICK(0))
129 IF STRIG(0)=0 THEN GOSUB 500
130 POKE 53248,P0:POKE 53249,P1:POKE 532
50,P2
135 POKE 656,2:POKE 657,5:? "VITESSE ";V
$;"  ":POKE 656,2:POKE 657,20:? "KILMS "
;KM;" "
140 CP=PEEK(53263):CS=PEEK(53255)
150 IF CP<>0 THEN 600
160 IF CS<>4 THEN 600
190 GOTO 90
210 DATA 64,228,238,238,238,238,78,68,22
9,245,255,255,233,255,255,233,255,255,24
5,229,68,78,238,238,238
215 DATA 238,238,64,-1
300 C=INT(RND(1)*3):S(C)=(RND(1)*2)+1:RE
TURN
400 P0=P0-(S(0)*2):IF P0<0 THEN P0=255
405 X=USR(ADR(E$),STICK(0))
410 P1=P1-(S(1)*2):IF P1<0 THEN P1=255
415 X=USR(ADR(E$),STICK(0))
420 P2=P2-(S(2)*2):IF P2<0 THEN P2=255
425 X=USR(ADR(E$),STICK(0))
430 KM=KM+0.05:RETURN
500 IF V$="VITE" THEN V$="LENT":SOUND 0,
255,12,10:SOUND 1,245,12,10:SOUND 2,235,
12,10:RETURN
510 V$="VITE":SOUND 0,100,12,10:SOUND 1,
110,12,10:SOUND 2,90,12,10:RETURN
600 GOTO 640
605 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,
0,0,0
610 POKE 53248,100:POKE 53249,100:POKE 5
3250,100
620 TRAP 620:? " UNE AUTRE  ";:INPUT V$:
IF V$(1,1)="O" THEN KM=0:V$="LENT":? CHR
$(125):GOTO 24
630 END
640 J=PEEK(203)+256*PEEK(204)
645 FOR Q=1 TO 250:IF PEEK(J+Q)=0 THEN N
EXT Q
650 ST=J+Q:VI=INT(ST/256):LT=ST-256*VI:P
OKE 203,LT:POKE 204,VI
655 FOR P=1 TO 60
660 X=USR(1536):SOUND 1,PEEK(53770),120,
15
670 POKE 707,PEEK(53770):NEXT P
700 POKE 53251,0:GOTO 605
1000 RESTORE 2000
1010 TRAP 1050:P=0
1020 P=P+1:READ A
1030 E$(P,P)=CHR$(A)
1040 GOTO 1020
1050 RESTORE 3000
1060 TRAP 1090:P=1536
1070 READ A:POKE P,A:P=P+1
1080 GOTO 1070
1090 TRAP 40000
1100 RETURN
2000 DATA 104,104,104,74,72,176,11,160,0
,177,203,136,145,203,200,200,208,247,104
,74,72
2010 DATA 176,11,160,0,177,203,200,145,2
03,136
2020 DATA 136,208,247,104,74,72,176,7,19
8,205,165
2030 DATA 205,141,3,208,104,74,176,7,230
,205,165,205,141,3,208,96,END
3000 DATA 104,160,0,173,10,210,145,203,2
00,192,25,208,246,96,END
```

Le jeu d'Adam

Avec l'Adam (une machine de 80 K de RAM, pourvue d'un programme de traitement de texte intégré, etc.), Coleco entre sur le marché de l'informatique familiale.

Le système Adam est construit autour de la console de jeux ColecoVision, et les propriétaires de cette console peuvent la transformer en un système complet en se procurant quelques unités complémentaires.

Au moment du déballage du système Adam, le nombre de composants est un peu surprenant. Le module ColecoVision se place sur un support de plastique, et le module de mémoire (qui renferme le microprocesseur Z80A et l'unité à cassette) est connecté à l'unité de jeu au moyen d'un port d'extension. Le module est fixé par des attaches de fixation sur le support. Le clavier est connecté au moyen d'un câble, et l'un des deux contrôleurs de jeux disponibles peut être enfiché sur le côté du clavier afin de servir de clavier numérique. L'imprimante est connectée au module de mémoire, et l'alimentation est fournie au système par l'intermédiaire d'un connecteur de secteur branché sur l'imprimante.

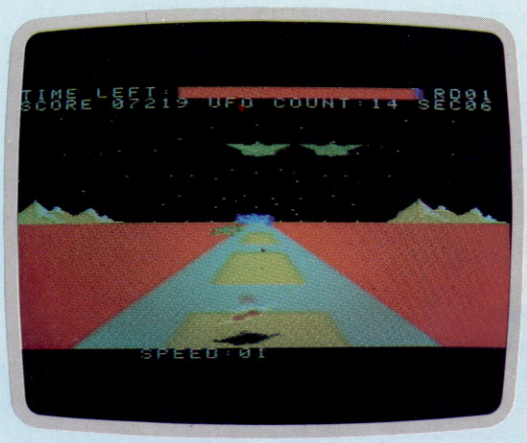
Le clavier a 75 touches, un ensemble de touches de traitement de texte, des touches de déplacement de curseur, et les touches alphanumériques habituelles.

Lors de la mise sous tension initiale du système, l'Adam est dans le mode « machine à écrire électronique ». Dès que vous appuyez sur une touche, un caractère est affiché à l'écran et est imprimé sur imprimante. L'écran représente une feuille de papier et un cylindre d'impression de machine à écrire. L'efficacité du système est limitée dans ce mode, mais il suffit d'appuyer sur une touche pour passer au mode « SmartWriter ».

SmartWriter est un programme de traitement de texte simple mais efficace qui est parfaitement conçu pour le débutant. L'affichage du cylindre d'impression de machine à écrire est conservé, et la ligne du bas de l'écran affiche les commandes attribuées aux touches de fonction. Une utilisation judicieuse de ces touches, ainsi que des touches de commande de traitement de texte du clavier, facilite beaucoup l'utilisation de SmartWriter. En fait, le manuel est superflu, puisque les options des menus guident l'utilisateur dans toutes les étapes nécessaires pour stocker, afficher et imprimer du texte. Le format de l'écran est 36 colonnes par 20 lignes, mais des lignes plus longues peuvent être créées en utilisant l'écran comme une fenêtre sur le texte entier.

Le lecteur de bande est de conception similaire au Microdrive de Sinclair, mais a une plus

La famille Adam
L'Adam est une machine « tout ou rien ». Il est vendu sous la forme d'un système complet : micro, lecteur de bande rapide, imprimante à marguerite, clavier, deux manches à balai et trois programmes (deux excellents jeux et un programme de traitement de texte). Cela fait de l'Adam un excellent ordinateur domestique, bien qu'il ne s'agisse que d'une console de jeu améliorée.
(Cl. Chris Stevens.)



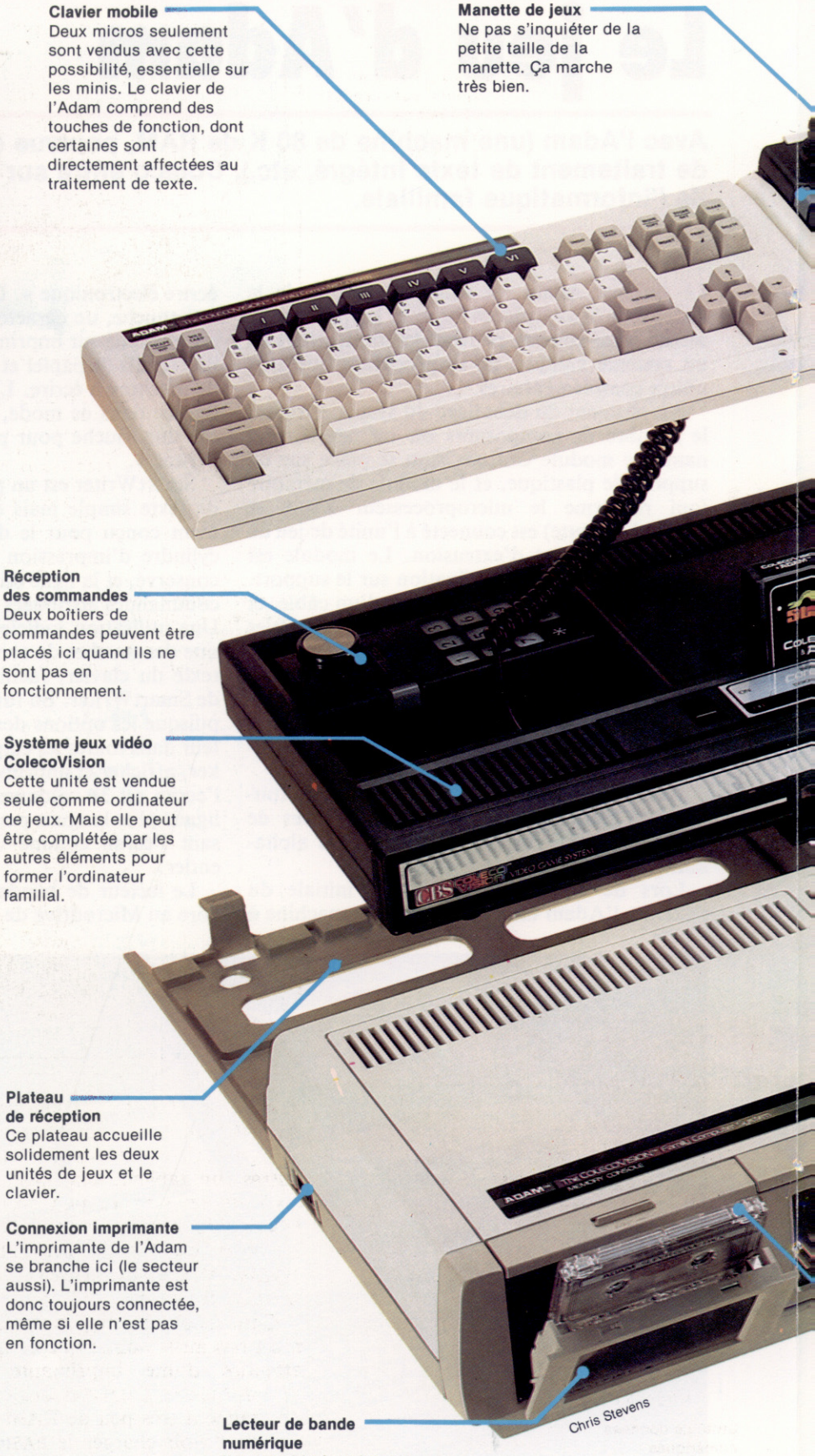
Buck Rogers
Le jeu Buck Rogers est livré avec le système Adam et utilise pleinement les possibilités du lecteur de bande. Le jeu est découpé en tranches pour des raisons de capacité mémoire. Il ne charge donc qu'une section du programme à la fois. A la fin de chaque jeu, la bande s'enroule automatiquement, et, si un pointage maximum a été réalisé, il est inscrit sur la bande. Par conséquent, le pointage maximum illustre le meilleur pointage jamais réalisé, et non seulement le meilleur pointage depuis la mise sous tension de l'ordinateur.

grande capacité (256 K par bande). Il est beaucoup plus rapide qu'une unité à cassette ordinaire. Il est aussi entièrement commandé par l'ordinateur; il est donc possible de rechercher un fichier particulier en rembobinant la bande. Les bandes fournies sont déjà formatées; il est donc impossible d'utiliser des bandes audio ordinaires. Lors de la mise sous tension de l'Adam, le programme de la bande placée dans le lecteur est chargé — il peut s'agir d'un module de jeu, d'une cassette SmartBASIC ou d'un programme conçu par l'utilisateur. Si le lecteur est vide, le système adopte le mode « machine à écrire électronique ».

L'imprimante à marguerite utilise du papier à pliage accordéon ou du papier ordinaire, et produit une qualité d'impression comparable à celle d'une machine à écrire électrique. Mais elle est extrêmement lente et bruyante.

Malgré les 80 K de RAM, SmartBASIC ne laisse que 28 K à l'utilisateur. SmartBASIC est comparable à la version Apple du BASIC; il est donc facile à utiliser, offre de bons graphiques et un choix de 16 couleurs. Cependant, le manuel BASIC est franchement mauvais et utilise un style plutôt douteux. Mais les similitudes avec le BASIC d'Apple créent une source de documentation très importante.

A première vue, le système Adam semble offrir un excellent rapport qualité/prix. Le logiciel intégré est idéal pour écrire des lettres, des rapports ou de courts articles et le système de bande est excellent. Mentionnons également que de nombreux jeux d'excellente qualité sont disponibles — non seulement les cartouches de jeux ColecoVision peuvent être utilisées sur l'Adam, mais l'achat d'un adapteur offre la possibilité d'exécuter la vaste gamme de jeux



Clavier mobile
Deux micros seulement sont vendus avec cette possibilité, essentielle sur les minis. Le clavier de l'Adam comprend des touches de fonction, dont certaines sont directement affectées au traitement de texte.

Manette de jeux
Ne pas s'inquiéter de la petite taille de la manette. Ça marche très bien.

Réception des commandes
Deux boîtiers de commandes peuvent être placés ici quand ils ne sont pas en fonctionnement.

Système jeux vidéo ColecoVision
Cette unité est vendue seule comme ordinateur de jeux. Mais elle peut être complétée par les autres éléments pour former l'ordinateur familial.

Plateau de réception
Ce plateau accueille solidement les deux unités de jeux et le clavier.

Connexion imprimante
L'imprimante de l'Adam se branche ici (le secteur aussi). L'imprimante est donc toujours connectée, même si elle n'est pas en fonction.

Lecteur de bande numérique
Il ressemble à un lecteur de cassette classique mais c'est une unité pour bande très rapide sous contrôle total de l'ordinateur.

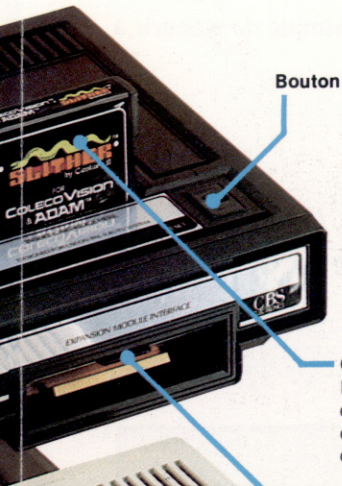
Chris Stevens

Chris Stevens

Boutons de tir

Clavier numérique
Les deux commandes de l'Adam se distinguent par leur clavier numérique intégré.

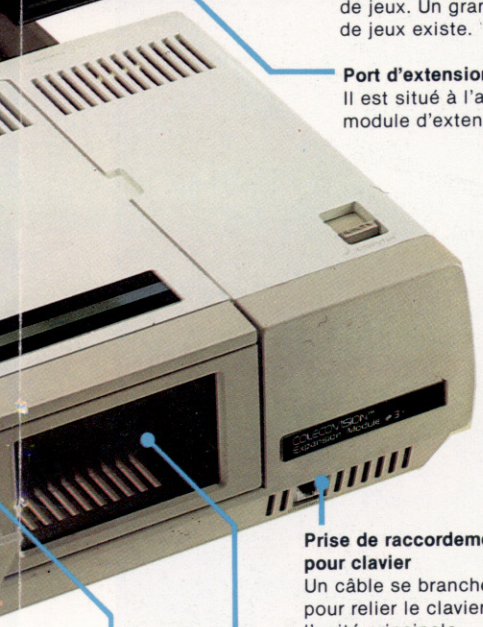
Support de commandes



Bouton remise à zéro

Cartouches de jeux
Elles s'enfichent directement dans l'unité de jeux. Un grand choix de jeux existe.

Port d'extension
Il est situé à l'arrière du module d'extension.



Prise de raccordement pour clavier
Un câble se branche ici pour relier le clavier à l'unité principale.

Espace pour un second lecteur de bande

Unité de données numériques
A ne pas confondre avec une cassette à bande classique. L'Adam utilise des bandes préformatées (256 K par bande).

Manches à balai de l'Adam
Deux manches à balai sont fournis avec la version de base de l'Adam. L'un d'entre eux sert de clavier numérique et se place contre le clavier. Coleco distribue également deux contrôleurs de jeu. Le Super Action Controller comprend une paire de manches à balai munis de claviers numériques et de gâchettes remplaçant les boutons de mise à feu normaux. Ceux-ci sont munis de connecteurs standard Atari de type D et peuvent être utilisés avec d'autres systèmes.
Le Roller Controller est un panneau de commande de jeu de type « arcades ». Il est muni de deux manches à balai Coleco standard et d'une boule roulante qui permet d'obtenir une réponse rapide et une commande précise. Deux ensembles de boutons de mise à feu autorisent la participation de deux joueurs, et un commutateur de mode permet de sélectionner le manche à balai ou la boule roulante.

Activision et Atari. Des programmes plus sérieux sont vendus sur bande, bien que leur nombre soit assez limité. On retrouve, entre autres, un tableur, une base de données, un outil de révision et une version du langage LOGO. L'addition du module d'extension de mémoire de 144 K permettra d'utiliser le système d'exploitation CP/M, et même des lecteurs de disquettes.

Côté inconvénients, la qualité d'impression n'est pas aussi bonne que ce que l'on pourrait attendre d'une imprimante à marguerite; l'imprimante est lente et bruyante. On dispose de vraiment très peu de RAM utilisateur, et le fait de devoir charger le BASIC à partir d'une bande devient rapidement assez gênant. Avec la récente baisse des prix des imprimantes à marguerite, il devrait être possible de composer un système doté de meilleures possibilités que l'Adam et coûtant le même prix.

Adam de Coleco

PRIX

**** (module d'extension)
***** (incluant la console de jeux ColecoVision).

DIMENSIONS

381 × 279 × 102 mm (unité d'extension de mémoire).
381 × 355 × 152 mm (imprimante).
381 × 152 × 51 mm (clavier).

UC

Z80A.

MÉMOIRE

80 K de RAM, dont 16 K sont utilisés par l'affichage vidéo. Peut atteindre 144 K avec un module de mémoire optionnel.

ÉCRAN

36 colonnes × 20 lignes (31 colonnes × 24 lignes en BASIC).
256 × 159 points en haute résolution avec 16 couleurs.

INTERFACES

Connecteur d'imprimante SmartWriter, fiche de téléphone modulaire Adam Net, 3 connecteurs d'extension, interface d'extension ColecoVision.

LANGAGES DISPONIBLES

SmartBASIC fourni sur cassette; système d'exploitation CP/M.

CLAVIER

75 touches, disposition QWERTY, touches moulées de type machine à écrire, 6 touches de fonction programmables.

DOCUMENTATION

Trois manuels accompagnent l'Adam; un manuel d'installation de 64 pages, un guide pour le programme de traitement de texte SmartWriter et un manuel de programmation SmartBASIC.
La documentation est généralement facile à suivre, mais incomplète. Le manuel BASIC est incompréhensible.

POINTS FORTS

Le choix abondant et l'excellente qualité des cartouches de jeu ColecoVision intéresseront les mordus des jeux sur ordinateur. Le programme de traitement de texte est facile à utiliser.

FAIBLESSES

Imprimante lente et bruyante. Modules de données disponibles uniquement chez Coleco, mais les logiciels sont limités.

Sur la pointe des pieds

Dans cette brève série d'articles, nous étudierons la réalisation d'un jeu graphique faisant usage du BBC BASIC. Il pourra tourner aussi bien sur le BBC Micro modèles A et B que sur l'Electron.

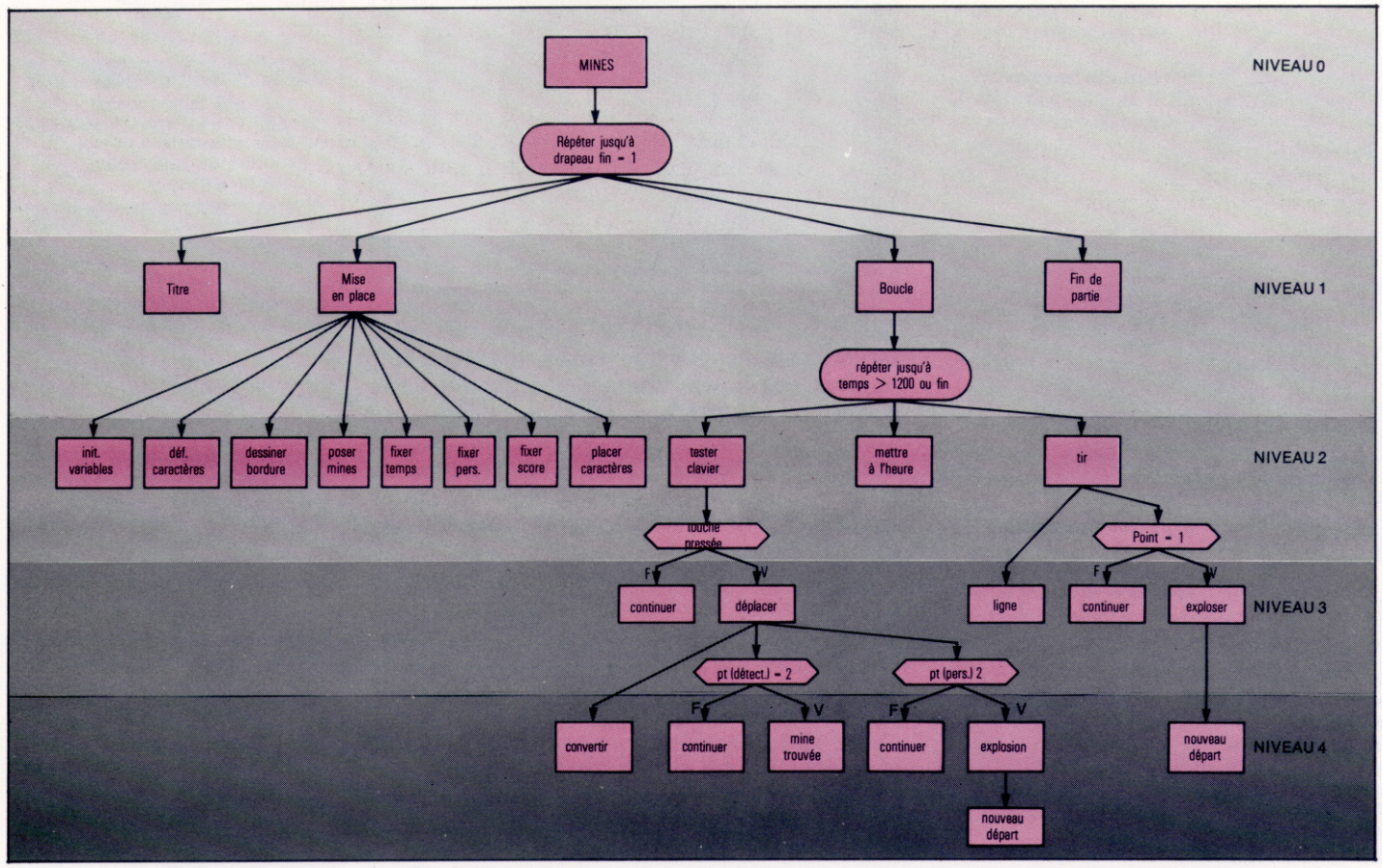
Pour le programmeur, le BASIC BBC a deux gros avantages sur le BASIC Microsoft standard : il est d'exécution rapide, et comporte de nombreux éléments qui permettent la programmation structurée. Celle-ci consiste essentiellement à rédiger des routines brèves, indépendantes entre elles, qu'on appelle parfois modules, et que l'on débogue séparément avant de les rassembler au sein du programme proprement dit. C'est naturellement une méthode de construction dont tout BASIC peut s'acquitter, mais celui du BBC Micro est doté de sous-routines spéciales appelées *procédures*. On pourrait les définir comme des éléments de programmation destinés à accomplir des tâches spécifiques.

Imaginons par exemple qu'il faille interrompre l'exécution après chaque instruction, et cela pour un laps de temps prédéterminé. En BASIC standard, on peut avoir recours à ce qu'on appelle une fausse boucle, qui ne fait rien, sinon demander un certain temps pour être accomplie :

```
10 PRINT "PREMIÈRE SECTION"
20 FOR I=1 TO 100 : NEXT I
30 PRINT "DEUXIÈME SECTION"
40 FOR I=1 TO 100 : NEXT I
50 PRINT "TROISIÈME SECTION"
60 FOR I=1 TO 100 : NEXT I
70 PRINT "QUATRIÈME SECTION"
80 END
```

Bien entendu, il serait plus simple de recourir à une sous-routine :

```
10  PRINT "PREMIÈRE SECTION"
20  GOSUB 100
30  PRINT "DEUXIÈME SECTION"
40  GOSUB 100
50  PRINT "TROISIÈME SECTION"
60  GOSUB 100
70  PRINT "QUATRIÈME SECTION"
80  END
100 REM ** SOUS-ROUTINE**
110 FOR I=1 TO 100 : NEXT I
120 RETURN
```



A la différence d'un ordinogramme, un diagramme structurel montre la façon dont les procédures sont utilisées, plutôt que le déroulement logique du programme. Une forme « en saucisse » indique une boucle de type REPEAT...UNTIL. Les losanges signalent les endroits où sont prises les décisions ; lorsque le test correspondant ne donne pas le résultat escompté, la boucle continue. Les niveaux soulignent la division du programme en blocs : chaque début de boucle, appel de procédure marque le début d'un bloc et correspond à un niveau logique moins élevé. (Cl. Janet Barrance.)

Ce procédé est remplacé dans le BBC BASIC par une procédure, qui donne le résultat suivant :

```
10 PRINT "PREMIÈRE SECTION"
20 PROCdélai
30 PRINT "DEUXIÈME SECTION"
40 PROCdélai
50 PRINT "TROISIÈME SECTION"
60 PROCdélai
70 PRINT "QUATRIÈME SECTION"
80 END
100 REM ** Définition de la procédure **
110 DEF PROCdélai
120 FOR I=1 TO 100 : NEXT I
130 ENDPROC
```

Mais, direz-vous, cela ne revient-il pas au même? Les deux méthodes font appel à une sous-routine placée après END, et qui peut être appelée de façon répétée à partir du programme! C'est exact; mais l'avantage de la procédure, c'est qu'elle est mise en œuvre par son nom, plutôt que par un numéro de ligne; la DÉFinition de la PROCédure elle-même peut être placée n'importe où après END.

Par ailleurs, si dans notre exemple nous voulions varier à chaque fois la durée de l'interruption, un nouvel avantage des procédures nous viendrait en aide : la possibilité d'assigner des valeurs à une définition de procédure. Imaginons par exemple qu'entre la première et la seconde section nous désirions faire un arrêt correspondant à 100 boucles, entre la deuxième et la troisième section à 200 boucles, et entre la troisième et la quatrième section à 175 boucles. En BASIC standard, il faudrait préciser à chaque fois la valeur de I avant tout appel de la sous-routine. Sur le BBC Micro, il suffit de placer la valeur correspondante entre parenthèses, chaque fois que la procédure est convoquée :

```
10 PRINT "PREMIÈRE SECTION"
20 PROCdélai (100)
30 PRINT "DEUXIÈME SECTION"
40 PROCdélai (200)
50 PRINT "TROISIÈME SECTION"
60 PROCdélai (175)
70 PRINT "QUATRIÈME SECTION"
80 END
100 REM ** DÉFINITION DE LA PROCÉDURE **
110 DEF PROCdélai(N)
120 FOR 1=1 TO N : NEXT 1
130 ENDPROC
```

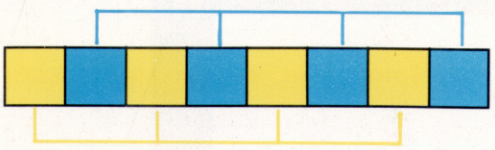
On peut également assigner plus d'une valeur à utiliser au sein de la procédure, si chacune est séparée des autres par des virgules. On peut aussi recourir à cette méthode pour mettre en œuvre des noms de variables qui correspondent à telle ou telle valeur.

Notre jeu est destiné à un seul joueur et se joue en appuyant sur les touches de contrôle du curseur. Vous déplacez un détecteur dans un champ de mines. Chacune de celles que vous désamorcez vous vaut des points. Mais il y a des obstacles. Le principal n'est autre que l'assistant qui vous accompagne : il imite consciencieusement chacun de vos mouvements et, lors de vos déplacements, il faut vous assurer que vous n'allez pas le faire buter sur une mine. De surcroît, on tire sur vous, et vous ne disposez que de deux minutes pour mener votre tâche à bien.

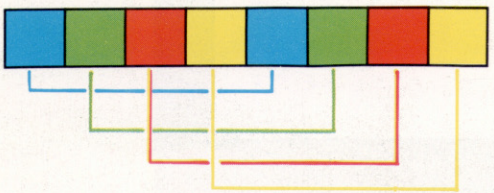
Le BBC BASIC faisant usage de procédures, les ordinogrammes ne sont pas très utiles. Un *diagramme structurel,* en revanche, permet de mieux voir quelles procédures sont nécessaires et comment elles sont conçues pour être appelées de l'intérieur du programme. Les boucles de type REPEAT...UNTIL ont dans notre diagramme une forme « en saucisse ». Les points du programme où s'effectuent des décisions ont une forme en losange plus classique mais, pour des raisons de place, nous en avons supprimé les extrémités supérieure et inférieure.

Avant même de placer des objets sur l'écran, il nous faut décider du choix d'un mode d'affichage écran. Trois facteurs principaux sont à prendre en compte : la résolution, la couleur et la mémoire. De façon générale, plus il faut brasser d'informations, plus il faut d'espace mémoire. Une résolution plus élevée et un plus grand nombre de couleurs demandent donc plus de mémoire. Si votre programme est court, cela n'a pas beaucoup d'importance, mais celui que nous voulons réaliser est assez long. Il nous faut plusieurs couleurs différentes pour distinguer les mines, le détecteur et l'assistant, et aussi pour donner au jeu un attrait visuel. Le BBC Modèle B propose deux modes « moyenne résolution ». Le mode 2 nous offre 16 couleurs, le mode 5 seulement 4. Il réclame beaucoup moins d'espace mémoire que le précédent, et un examen de la façon dont les informations sont stockées va nous montrer pourquoi.

A la différence d'autres ordinateurs, le BBC stocke en un seul octet, pour une petite zone donnée de l'écran, les informations relatives à la couleur et à l'« allumage » et à l'« extinction » de chaque pixel. En mode 2, il faut 4 bits pour représenter les 16 couleurs possibles. Un octet ne peut donc contenir que l'information concernant la couleur de deux pixels seulement. Voici la disposition des bits dans chaque octet :

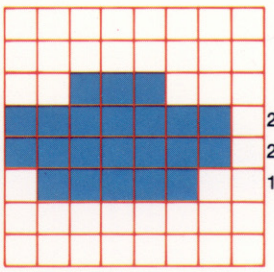


Le mode 5 se limitant à 4 couleurs, 2 bits suffiront à détenir les informations à ce sujet. Un octet représente donc 4 pixels.

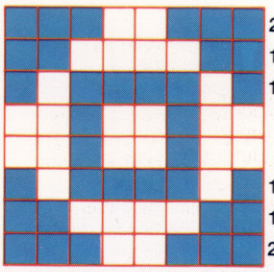


Les deux modes ont une résolution de 160 × 256 pixels. Le nombre d'octets dont le mode 2 a besoin est donc de (160 × 256)/2 = 20 K, le

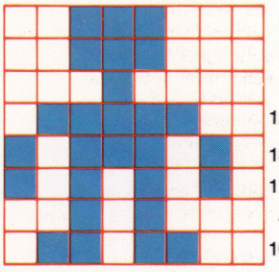
Mine



Détecteur



Assistant



L'image et ses points
Ces caractères redéfinis représentent la mine, le détecteur et l'assistant. L'information relative aux pixels qui les composent est à chaque fois répartie sur 8 octets, dont chacun est l'« image », binaire d'une rangée de la forme correspondante.

mode 5 se contentant de (160 × 256)/4 = 10 K de mémoire écran. Le choix du mode 5 nous donne donc 10 K de mémoire supplémentaire. De surcroît, il est accessible sur les modèles A du BBC, tandis que le mode 2 ne l'est pas !

La commande VDU23 permet au programmeur de redéfinir des caractères, les codes ASCII correspondants étant compris entre 226 et 225. Le nombre qui suit immédiatement la commande VDU23 indique quel code on assigne à la forme définie par les huit nombres qui viennent ensuite.

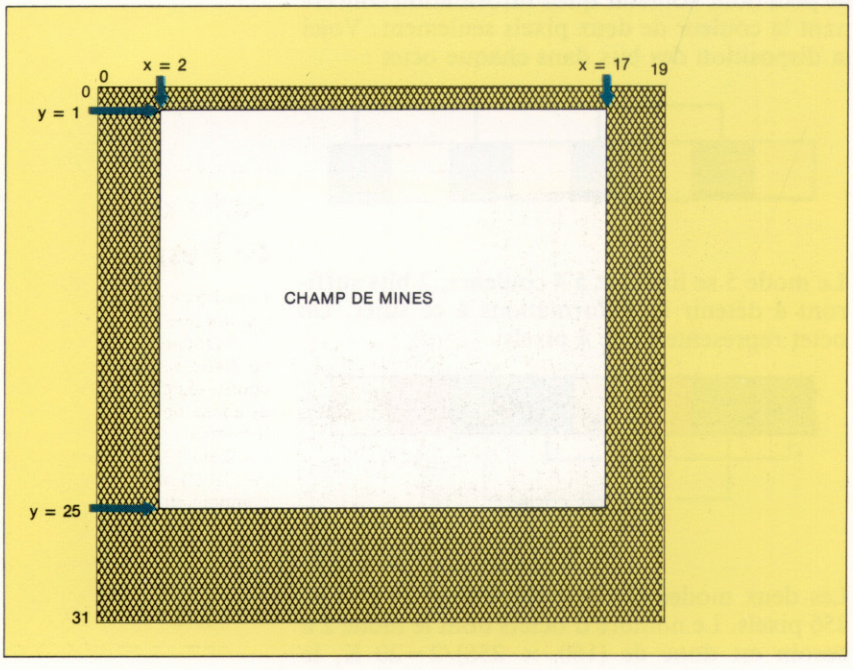
```
VDU23,224,0,0,56,254,254,124,0,0
```

fait de CHR$(224) une mine. Pour l'imprimer sur l'écran il suffit de recourir à PRINT CHR$(224). La procédure suivante permet de définir les trois caractères utilisés dans le jeu :

```
2380 DEF PROCdéfinition caractères
2390 REM ** MINES **
2400 VDU23,224,0,0,56,254,254,124,0,0
2410 REM ** DETECTEUR DE MINES **
2420 VDU23,225,231,195,189,36,36,189,195,231
2430 REM ** ASSISTANT **
2440 VDU23,226,56,56,16,124,186,170,40,108
2450 ENDPROC
```

Une fois les formes définies, nous allons pouvoir les imprimer sur l'écran, le plus facile étant d'utiliser la commande PRINT TAB(X,Y). En mode 5, nous disposons de 32 lignes de 20 caractères, ce qui veut dire que X va de 0 à 19, et *y* de 0 à 31. Le champ de mines occupera la position indiquée sur le diagramme.

Pour y répartir les mines au hasard, faisons usage de la commande RND(N). Si N est un nombre entier, nous obtiendrons un nombre entier compris entre 1 et N. La coordonnée horizontale de chaque mine doit être comprise entre 2 et 17. RND(16) nous donne un nombre entre 1 et 16, dont RND(16)+1 nous permettra d'avoir des coordonnées satisfaisantes pour ce que nous voulons :



Une zone dangereuse
Le champ de mines est réparti sur vingt lignes de seize caractères, et les mines y sont « semées » au hasard pendant la procédure d'initialisation. (Cl. Janet Barrance.)

```
2560 DEF PROCpose mines (nombre mines)
2570 REM ** FAIT PASSER LA COULEUR 2 AU VERT **
2580 VDU19,2,2,0,0,0,
2590 FOR I=1 TO nombre mines
2600 PRINTTAB(RND(16)+1,RND(25));CHR$(224)
2610 NEXT I
2620 ENDPROC
```

En mode 5, nous sommes limités à quatre couleurs — ordinairement le noir, le rouge, le jaune et le bleu, qui correspondent respectivement aux numéros de couleurs 0, 1, 2 et 3. Mais nous ne sommes pas contraints d'en rester là, grâce à l'emploi de la commande VDU19. Les nombres 0 à 3 sont ce qu'on appelle les couleurs *logiques*. Chacune des seize nuances du BBC possède un autre numéro, indépendant du mode d'affichage utilisé. C'est ce qu'on appelle les couleurs *réelles,* dont vous trouverez la table dans le manuel utilisateur. Chacune des quatre couleurs logiques peut prendre l'une des seize couleurs réelles. Nous souhaitons que dans notre jeu les mines soient vertes (couleur réelle numéro 2) et non jaunes (couleur logique numéro 2). La commande VDU19 permet d'effectuer ce changement.

Le détecteur et l'assistant sont respectivement situés, en début de partie, en bas à gauche et en haut à droite de l'écran. Comme nous aurons sans doute besoin de les replacer ailleurs ultérieurement, la procédure nécessaire fera donc usage de variables, et nous aurons donc à nous servir de xdét,ydét et de xass,yass.

```
2830 DEF PROCposition caractères
2840 COLOUR 1
2850 PRINTTAB(xdet,ydet);CHR$(225)
2860 PRINTTAB(xass,yass);CHR$(226)
2870 COLOUR 2
2880 ENDPROC
```

La commande COLOUR, placée au début et à la fin de la procédure, détermine la couleur logique qui sera attribuée au texte. COLOUR 1 sélectionne la couleur logique 1 (rouge) pour le détecteur et l'assistant, tandis que COLOUR 2 revient à la couleur verte pour toute impression ultérieure. Avant tout usage de cette procédure, cependant, il faudra assigner des valeurs aux variables xdét, ydét, xass et yass. Il faut pour cela une autre procédure, tandis que d'autres variables indispensables devront être initialisées :

```
2320 DEF PROCinitialisation variables
2330 xdet=2 : ydet=25 : xass=17 : yass=1
2340 xdéb=120 :xfin=1144
2350 zéro$="000000"
2360 ENDPROC
```

Toutes ces procédures peuvent être testées grâce à un petit programme d'appel :

```
 5 MODE 5
10 COLOUR 2
20 PROCinitialisation-variables
30 PROCdéfinition-caractères
40 PROCpose-mines (40)
50 PROCposition-caractères
60 END
```

Dans l'article suivant, nous verrons comment introduire le facteur temporel et contrôler le mouvement à partir du clavier.

Mécano de l'espace

Jet Pac a marqué les débuts d'une firme qui devait faire parler d'elle : Ultimate, Play The Game. Écrit au départ pour Spectrum, une version destinée au Vic-20 est vite apparue.

On méprise volontiers ces jeux où il faut exterminer le plus possible d'extra-terrestres. Pourtant, en rédiger un qui soit vraiment réussi est une tâche qui demande un grand talent de programmeur. Jet Pac en est un bon exemple.

Chef pilote de l'« Acme Interstellar Transport Company », vous devez voyager à travers la galaxie pour assembler des navires spatiaux sur les planètes où cela est nécessaire, tout en recueillant autant d'or et de pierres précieuses que vous le pourrez. L'Hydrovac Jet Pac vous aide grandement dans votre travail de construction. Il peut soulever et déplacer à peu près n'importe quoi, et vous permet de manipuler à votre aise les diverses composantes des astronefs. Vos lasers photoniques ramèneront à la raison tout extra-terrestre assez vulgaire pour se plaindre de la pollution que vous répandez sur sa planète.

La description du jeu fait penser à une grandiose épopée en Technicolor, une équipée à travers toutes les planètes de la galaxie, mais, bien entendu, les choses sont plus terre à terre. Tous les lieux que vous visitez sont à peu près identiques, et le héros doit s'ennuyer un peu. Mais heureusement, il y a les extra-terrestres. Chaque planète est habitée par une seule race — ce qui n'a rien d'étonnant, ces gens-là paraissant se reproduire comme des lapins. Chaque espèce a une forme bien particulière, de la soucoupe volante au ballon qui rebondit, mais toutes ont un point commun : les toucher entraîne une mort immédiate.

Le vaisseau spatial est formé de trois parties, dispersées sur la surface de la planète, que vous devez rassembler avant de pouvoir vous envoler pour un autre endroit. Pas besoin de tournevis : il suffit de ramener les éléments à la base, et l'astronef se construit sous vos yeux.

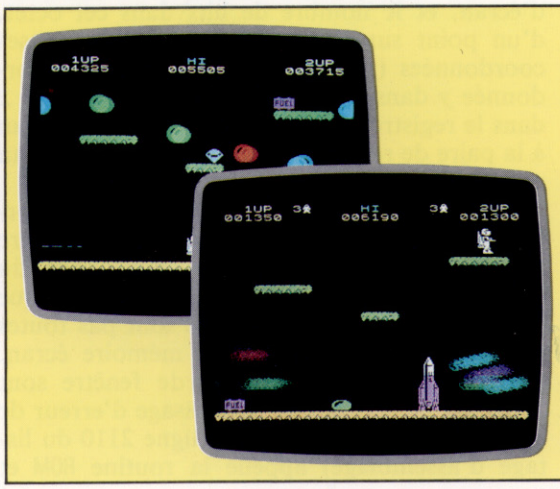
Les extra-terrestres font tout l'intérêt du jeu. Ils sont très nombreux et donnent d'abord l'impression de se déplacer au hasard, se dirigeant vers vous de bien menaçante façon. Mais vous vous rendez compte au bout d'un certain temps qu'ils suivent un chemin prédéterminé, dont seul le point de départ est aléatoire. La première vague d'assaillants descend très lentement, et vous avez tout le temps de les abattre avec vos lasers. La seconde est constituée de balles qui rebondissent à travers l'écran entre le sol et les corniches rocheuses. Ce mélange de hasard et de trajectoires fixes est parfaitement réussi, et le jeu demande beaucoup d'adresse et des réflexes rapides.

Un choix malencontreux des touches de contrôle suffit souvent à faire perdre tout intérêt à ce type de jeu, mais ce n'est pas le cas ici. Deux touches placées tout en bas du clavier permettent un déplacement à droite ou à gauche. La seconde rangée de touches déclenche les lasers (qui disposent d'une option tir continu). La troisième met en marche les réacteurs qui vous font monter, tandis que la dernière vous fait planer. Il ne faut pas oublier un petit détail qui est très réussi : si vous n'appuyez sur aucune touche, vous redescendrez peu à peu vers le sol, en raison de la gravité; nous entendons ici par « gravité » la force d'attraction de la planète!

Le graphisme est excellent. Le pilote lui-même est très réussi, tout comme l'Hydrovac Jet Pac, qui lâche des nuées de fumée très convaincantes quand il est mis en marche. Les rayons lasers tracent dans le ciel des lignes multicolores, et les extra-terrestres, une fois touchés, explosent et partent en fumée.

Le jeu est pratiquement le même sur le Spectrum et sur le Vic-20; l'affichage de ce dernier fait que le pilote a l'air un peu obèse. Mais dans les deux cas le jeu est tout à fait passionnant.

Jet Pac : pour le Spectrum; 16 ou 48 K; pour le Vic-20 8 K.
Éditeurs : Ashby Computers and Graphics, Ltd., The Green Ashby de la Zouch, Leicestershire, LE6 5JU.
Auteurs : Ultimate, Play the Game.
Manettes : Kempston Competition Pro (Spectrum). La plupart des manettes Commodore (Vic-20).
Format : cassette.



Bâtisseur de l'espace

Les pilotes de Jet Pac, qui ont pour objectif d'assembler les vaisseaux spatiaux, se voient contrariés par les attaques des extra-terrestres; mais l'or et les diamants tombant du ciel (!) offrent des compensations.
(CL. Liz Heaney.)

Jet Pac sur le Spectrum

Fenêtre sur écran

Créer des « fenêtres » sur un écran de visualisation est particulièrement utile. Nous examinons à ce propos un programme en langage machine pour Spectrum.

Ouvrir une fenêtre
Le mot « HELLO » défile horizontalement et verticalement sur la fenêtre découpée sur l'écran, pixel par pixel.

Notre programme en code machine vous permet de définir des fenêtres rectangulaires sur l'écran du Spectrum, et de les faire défiler horizontalement et verticalement. Les fenêtres sont de taille quelconque et elles peuvent être placées en n'importe quel endroit de l'écran; il n'est pas nécessaire qu'elles occupent des carrés de caractères de 8 × 8 pixels.

Ce programme utilise des tableaux commençant à l'adresse $B004 (45060 décimal) pour le maniement des paramètres de fenêtre et pour le stockage temporaire de données. Il prend l'adresse du tableau pour la fenêtre voulue à partir des adresses $B000 et $B001 (54056 et 45057 décimaux). Pour chaque fenêtre, le tableau requiert 11 octets; donc, s'il faut plus d'une fenêtre, le tableau pour la seconde fenêtre commencera en $B00F (45071 décimal), le tableau pour la troisième en $B01A (45082 décimal), etc.

Le programme de démonstration BASIC n'utilise qu'une fenêtre. Les détails de cette fenêtre sont entrés en mémoire par POKE aux lignes 180-230, et la routine d'initialisation de fenêtre est appelée à la ligne 240. S'il faut plus d'une fenêtre, chacune d'elles doit être définie ainsi avant de pouvoir être utilisée. Pour modifier des fenêtres, il faut POKEr l'adresse du tableau de la nouvelle fenêtre dans les emplacements mémoire WT et WT+1. Les directions de défilement sont données en entrant les valeurs (POKE) dans les emplacements mémoire WNDWTB+DIR — il suffit de POKEr 0 pour le défilement vers la gauche, 1 vers la droite, 2 vers le haut, ou 3 vers le bas.

Le programme en langage d'assemblage commence par définir un nombre de constantes. PIXADR est un sous-programme dans la ROM du Spectrum qui calcule l'adresse de l'octet d'écran, et le nombre de bits dans cet octet, d'un point sur l'écran qui est défini par ses coordonnées (par PLOT). PIXADR prend la coordonnée *y* dans le registre B et la coordonnée *x* dans le registre C, puis transmet l'adresse écran à la paire de registres HL et la position de bit au registre A.

La routine INITW vérifie d'abord que les coordonnées pour le coin inférieur droit de la fenêtre sont bien en bas et à droite des coordonnées du coin supérieur gauche, et assure également que les marges gauche et droite ne sont pas toutes deux dans le même octet de mémoire écran.

Les erreurs d'initialisation de fenêtre sont affichées par la routine de message d'erreur de la ROM. L'instruction RST 8 (ligne 2110 du listage d'assemblage) appelle la routine ROM et retourne au mode BASIC, et DEFB 25 à la ligne 2120 fournit le message « Erreur de paramètre Q ».

La dernière section de INITW calcule LEFTMSK et RTMASK, qui sont utilisés lorsqu'on fait défiler les octets d'écran aux marges de la fenêtre, certaines parties d'octet pouvant être à l'intérieur et d'autres à l'extérieur de la fenêtre. Les bits individuels des masques qui correspondent aux bits d'écran à l'extérieur des marges de fenêtre sont mis à un, tandis que ceux correspondant aux bits intérieurs sont mis à zéro.

Le programme de défilement commence réellement au label SCROLL. Le programme teste la direction de défilement et appelle HORIZ pour le défilement vers la gauche ou la droite, ou VERT pour le défilement vers le haut ou le bas.

Les défilements à gauche et à droite sont très semblables dans leur traitement; aussi, au lieu d'utiliser deux routines distinctes, les avons-nous combinées en une seule. Le code exact pour chaque direction est sélectionné en testant le bit zéro de l'octet de direction. Pour voir comment marche HORIZ, nous allons examiner le défilement à gauche.

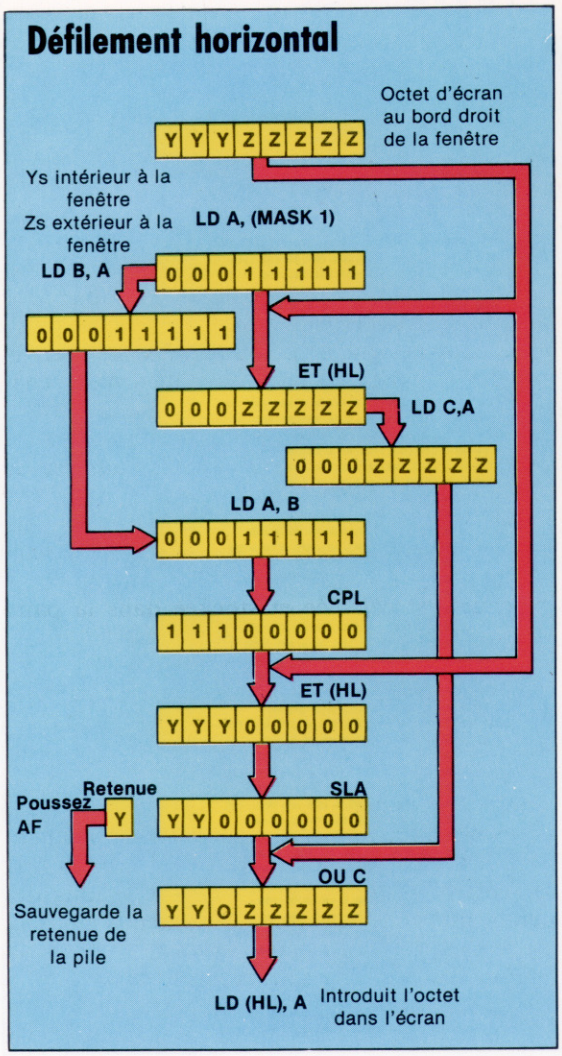
Les défilements à gauche et à droite commencent tous deux par la ligne supérieure de pixels dans la fenêtre, puis descendent, de sorte que HORIZ commence par copier la coordonnée *y* pour la ligne supérieure dans la mémoire temporaire pour la ligne en cours. En défilant vers la gauche, nous devons partir du côté droit de chaque ligne de pixels dans la fenêtre et aller vers la gauche. A cette fin, on copie RMASK et LMASK respectivement dans MASK1 et MASK2, l'adresse de l'octet d'écran à l'extrémité gauche de la ligne en cours de pixels est calculée et stockée dans la paire de registres DE, et l'adresse de l'octet d'écran à l'extrémité droite de la ligne de pixels en cours est calculée et stockée dans la paire de regitres HL. Le sous-programme HLNSCR est ensuite appelé pour faire défiler la ligne de pixels. La routine teste si la rangée inférieure de la fenêtre a été atteinte, sinon elle prend la ligne de pixels suivante et retourne en HORIZ3 pour continuer le défilement.

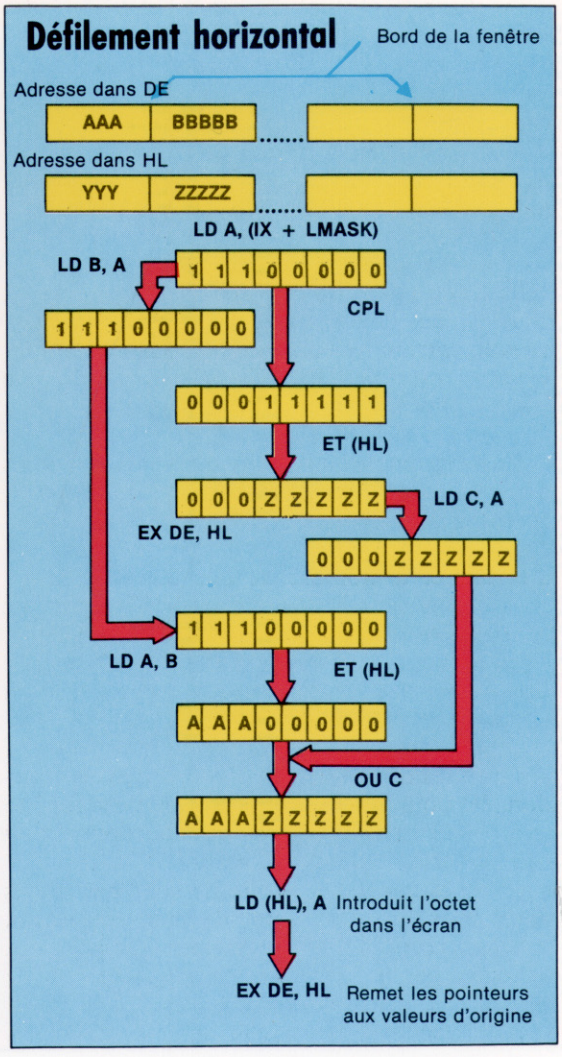
HLNSCR commence à une extrémité de la ligne de pixels, avec un octet, dont certains bits peuvent être à l'intérieur et d'autres à l'extérieur de la fenêtre. Ils se déplacent ensuite le long des octets qui sont entièrement contenus dans la fenêtre. HLNSCR s'achève à l'autre marge de la fenêtre, où également certains bits peuvent être dans et certains hors de la fenêtre. Nous illustrons cela par un schéma montrant comment

opère le code sur l'octet de droite lors du défilement à gauche. La section de HLNSCR commençant en NEXT fait défiler les octets à l'intérieur de la fenêtre. Le bit, qui est sorti de l'octet précédent pour entrer dans le drapeau de retenue, a été sauvegardé sur la pile par une instruction PUSH AF, et est remis dans le drapeau de retenue par POP AF. Pour un défilement vers la gauche, la routine choisit l'instruction RL (HL) et l'octet sur l'écran est décalé à gauche, le bit du drapeau de retenue étant déplacé à gauche de l'octet et le bit de droite entrant dans le drapeau de retenue. PUSH AF sauvegarde le drapeau de retenue, de sorte que ce bit peut être déplacé dans l'octet suivant. Pour tester la fin de la ligne, la routine compare simplement les registres L et E; cela parce que l'octet HI d'une adresse écran sera la même pour tous les octets d'écran de la même ligne. Le défilement partiel du dernier octet se termine de la même façon que pour le premier octet.

Les routines de défilement vertical sont combinées d'une manière analogue. Si nous examinons ce qui se passe lorsque VERT défile vers le haut, nous voyons que la routine commence par copier la coordonnée *y* pour la ligne supérieure dans un stockage temporaire pour la ligne en cours. Les adresses écran des octets de marges gauche et droite dans la ligne sont ensuite calculées et la longueur de ligne déterminée. La routine met l'adresse de l'octet de marge gauche dans DE et l'adresse de l'octet correspondant de la ligne immédiatement en dessous, dans HL, avant d'appeler le sous-programme VLNSCR pour faire le défilement. Ensuite, VERT teste s'il a atteint le bas de la fenêtre. Sinon, il descend d'une ligne avant de retourner en VERT5 pour faire défiler une autre ligne de pixels. Si le bas a été atteint, la section de routine commençant en CLREDG remplit la ligne de pixels du bas avec des zéros pour effacer la ligne sur l'écran.

VLNSCR traite séparément les octets de marge de la même manière que HLNSCR. La façon dont le code traite ces octets extrêmes est illustrée dans notre second schéma. Pour déplacer la partie centrale de la ligne de pixels, la routine incrémente HL et DE afin qu'ils pointent vers le premier octet intérieur de la ligne en cours et l'octet correspondant de la ligne du dessus. VLNSCR calcule ensuite la longueur de la section centrale (les octets qui tombent entièrement à l'intérieur de la fenêtre), charge cette information dans BC et utilise l'instruction de déplacement de bloc LDIR pour déplacer toute la section centrale de cette ligne de la fenêtre d'une ligne vers le haut.





Descendre la ligne
Le défilement horizontal présente des problèmes particuliers au bord de la fenêtre : l'octet d'écran doit être masqué pour isoler les bits de pixels à l'intérieur et à l'extérieur de la fenêtre, et le contenu de l'octet doit être décalé. Les deux processus emploient ET, OU et SHIFT, et la pile est utilisée pour sauvegarder le PSR.
Le défilement vertical est rendu plus simple par la table d'implantation mémoire de l'écran du Spectrum. L'octet d'écran doit être masqué pour isoler les pixels à l'intérieur et à l'extérieur de la fenêtre,
et l'instruction EX est utilisée pour échanger les contenus des registres DE et HL, qui contiennent les pointeurs d'adresses écran.

Fenêtres sur le Spectrum

Instruction de traitement

1. Taper le programme de démonstration BASIC.
2. Sauvegarder (SAVE «SCROLL» LINE 5).
3. Taper et exécuter (RUN) le programme chargeur langage machine.
4. Sauvegarder (SAVE «SCROLLMC» CODE 45312, 410) directement sur cassette après le programme BASIC.
5. Rembobiner la cassette et charger (LOAD «SCROLL»).

Listage d'assemblage

```
 10 PIXADR EQU  #22AA
 20 WT     EQU  #8000
 30 MASK1  EQU  #8002
 40 MASK2  EQU  #8003
 50 WNDWTB EQU  #8004
 60 LEFTX  EQU  0
 70 TOPY   EQU  1
 80 RIGHTX EQU  2
 90 BOTY   EQU  3
100 LBIT   EQU  4
110 RBIT   EQU  5
120 CURNTY EQU  6
130 DIR    EQU  7
140 LMASK  EQU  8
150 RMASK  EQU  9
160 LENGTH EQU  10
170        ORG  #B100
180 INIT   LD   HL,(WT)
190        PUSH HL
200        POP  IX
210        CALL INITW
220        RET
230 SCROLL LD   HL,(WT)
240        PUSH HL
250        POP  IX
260        BIT  1,(IX+DIR)
270        PUSH AF
280        CALL Z,HORIZ
290        POP  AF
300        CALL NZ,VERT
310        RET
320 HORIZ  LD   A,(IX+TOPY)
330        LD   (IX+CURNTY),A
340        BIT  0,(IX+DIR)
350        JR   Z,HORIZ1
360        LD   B,(IX+LMASK)
370        LD   A,(IX+RMASK)
380        JR   HORIZ2
390 HORIZ1 LD   B,(IX+RMASK)
400        LD   A,(IX+LMASK)
410 HORIZ2 LD   (MASK2),A
420        LD   A,B
430        LD   (MASK1),A
440 HORIZ3 LD   C,(IX+LEFTX)
450        LD   B,(IX+CURNTY)
460        CALL PIXADR
470        EX   DE,HL
480        LD   C,(IX+RIGHTX)
490        LD   B,(IX+CURNTY)
500        CALL PIXADR
510        BIT  0,(IX+DIR)
520        JR   Z,HORIZ4
530        EX   DE,HL
540 HORIZ4 CALL HLNSCR
550        LD   A,(IX+BOTY)
560        CP   (IX+CURNTY)
570        RET  Z
580        DEC  (IX+CURNTY)
590        JR   HORIZ3
600 HLNSCR LD   A,(MASK1)
610        LD   B,A
620        AND  (HL)
630        LD   C,A
640        LD   A,B
650        CPL
660        AND  (HL)
670        BIT  0,(IX+DIR)
680        JR   Z,HLN1
690        SRA  A
700        JR   HLN2

710 HLN1   SLA  A
720 HLN2   PUSH AF
730        OR   C
740        LD   (HL),A
750 NEXT   BIT  0,(IX+DIR)
760        JR   Z,HLN3
770        INC  HL
780        JR   HLN4
790 HLN3   DEC  HL
800 HLN4   LD   A,L
810        CP   E
820        JR   Z,LAST
830        POP  AF
840        BIT  0,(IX+DIR)
850        JR   Z,HLN5
860        RR   (HL)
870        JR   HLN6
880 HLN5   RL   (HL)
890 HLN6   PUSH AF
900        JR   NEXT
910 LAST   LD   C,(HL)
920        LD   A,(MASK2)
930        AND  C
940        LD   B,A
950        POP  AF
960        BIT  0,(IX+DIR)
970        JR   Z,HLN7
980        RR   C
 990        JR   HLN8
1000 HLN7  RL   C
1010 HLN8  LD   A,(MASK2)
1020       CPL
1030       AND  C
1040       OR   B
1050       LD   (HL),A
1060       PET
1070 VERT  LD   C,(IX+LEFTX)
1080       BIT  0,(IX+DIR)
1090       JR   Z,VERT1
1100       LD   B,(IX+BOTY)
1110       JR   VERT2
1120 VERT1 LD   B,(IX+TOPY)
1130 VERT2 LD   (IX+CURNTY),B
1140       CALL PIXADR
1150       PUSH HL
1160       PUSH HL
1170       LD   C,(IX+RIGHTX)
1180       LD   B,(IX+CURNTY)
1190       CALL PIXADR
1200       POP  DE
1210       AND  A
1220       SBC  HL,DE
1230       LD   A,L
1240       DEC  A
1250       LD   (IX+LENGTH),A
1260 VERT3 BIT  0,(IX+DIR)
1270       JR   Z,VERT4
1280       INC  (IX+CURNTY)
1290       JR   VERT5
1300 VERT4 DEC  (IX+CURNTY)
1310 VERT5 LD   C,(IX+LEFTX)
1320       LD   B,(IX+CURNTY)
1330       CALL PIXADR
1340       POP  DE
1350       PUSH HL
1360       CALL VLNSCR
1370       LD   A,(IX+CURNTY)
1380       BIT  0,(IX+DIR)
1390       JR   Z,VERT6
1400       CP   (IX+TOPY)
1410       JR   VERT7
1420 VERT6 CP   (IX+BOTY)
1430 VERT7 JR   NZ,VERT3
1440 CLREDG POP  HL
1450       LD   A,(IX+LMASK)
1460       AND  (HL)
1470       LD   (HL),A
1480       LD   B,(IX+LENGTH)
1490       LD   A,0
1500 CLR1  INC  HL
1510       LD   (HL),A
1520       DJNZ CLR1
1530       INC  HL
1540       LD   A,(IX+RMASK)
1550       AND  (HL)
1560       LD   (HL),A
1570       RET
1580 VLNSCR LD  A,(IX+LMASK)
1590       CALL ENDBYT
1600       INC  HL
1610       INC  DE
1620       LD   B,0
1630       LD   C,(IX+LENGTH)
1640       LDIR
1650       LD   A,(IX+RMASK)
1660 ENDBYT LD  B,A
1670       CPL
1680       AND  (HL)
1690       LD   C,A
1700       EX   DE,HL
1710       LD   A,B
1720       AND  (HL)
1730       OR   C
1740       LD   (HL),A
1750       EX   DE,HL
1760       RET
1770 INITW LD   A,(IX+RIGHTX)
1780       CP   (IX+LEFTX)
1790       JR   Z,ERROR
1800       JR   C,ERROR
1810       LD   A,(IX+TOPY)
1820       CP   (IX+BOTY)
1830       JR   Z,ERROR
1840       JR   C,ERROR
1850       LD   C,(IX+LEFTY)
1860       LD   B,(IX+TOPY)
1870       CALL PIXADR
1880       PUSH HL
1890       LD   (IX+LBIT),A
1900       LD   C,(IX+RIGHTX)
1910       LD   B,(IX+TOPY)
1920       CALL PIXADR
1930       LD   (IX+RBIT),A
1940       POP  BC
1950       LD   A,C
1960       CP   L
1970       JR   Z,ERROR
1980 LFTMSK LD  B,(IX+LBIT)
1990       LD   A,0
2000 L1    SCF
2010       RRA
2020       DJNZ L1
2030       LD   (IX+LMASK),A
2040 RTMASK LD  B,(IX+RBIT)
2050       LD   A,255
2060 L2    AND  A
2070       RRA
2080       DJNZ L2
2090       LD   (IX+RMASK),A
2100       RET
2210 ERROR RST  B
2120       DEFB 25
```

Programme de démonstration BASIC

```
  5 CLEAR  32767
 10 LOAD ""CODE
 20 LET WT=45056
 30 LET WINDOWTABLE=45040: REM B004 HEX
 40 LET LEFTX=0
 50 LET TOPY=1
 60 LET RIGHTX=2
 70 LET BOTY=3
 75 LET DIR=7
 80 LET SCROLL=45322
 90 LET INIT=45312
100 BORDER 6
110 PAPER 4: INK 2
120 CLS
130 POKE WT,4: POKE WT+1,176
190 REM WT : WT+1 CONTIENNENT MAINTENANT L'ADRESSE 45060
IN LO,HI FORMAT
200 POKE WINDOWTABLE+LEFTX,5
210 POKE WINDOWTABLE+TOPY,80
220 POKE WINDOWTABLE+RIGHTX,250
230 POKE WINDOWTABLE+BOTY,35
240 RANDOMIZE USR INIT
250 FOR Y=0 TO 175
260 PLOT 0,Y
270 DRAW 244,0
280 NEXT Y
290 POKE WINDOWTABLE+DIR,2
300 RANDOMIZE USR SCROLL
320 NEXT Y
400 PRINT AT 12,20;"HELLO";
410 POKE WINDOWTABLE+DIR,0
420 FOR I=1 TO 138
430 RANDOMIZE USR SCROLL
440 NEXT I
470 POKE WINDOWTABLE+DIR,3
480 FOR I=1 TO 35
490 RANDOMIZE USR SCROLL
500 NEXT I
510 POKE WINDOWTABLE+DIR,1
520 FOR I=1 TO 130
530 RANDOMIZE USR SCROLL
540 NEXT I
550 POKE WINDOWTABLE+DIR,2
560 FOR I=1 TO 35
570 RANDOMIZE USR SCROLL
580 NEXT I
590 GO TO 410
999 STOP
```

Chargeur langage machine

```
100 > LET a=45312
110 FOR I=1000 TO 1500 STEP 10
120 LET s=0
130 READ b
140 POKE a,b
150 LET s=s+b
160 NEXT a
170 READ b
180 IF s<>b THEN PRINT "ERREUR A LA LIGNE";L: STOP
190 NEXT I
1000 DATA 42,0,176,229,221,225,205,69,1167
1010 DATA 178,201,42, ,176,229,221,225,1272
1020 DATA 221,203,7,78,245,204,29,177,1164
1030 DATA 241,196,184,177,201,221,126,1,1347
1040 DATA 221,119,6,221,203,7,70,40,887
1050 DATA 8,221,70,8,221,126,9,24,687
1060 DATA 6,221,70,9,221,126,8,50,711
1070 DATA 3,176,120,50,2,176,221,78,826
1080 DATA 0,221,70,6,205,170,34,235,941
1090 DATA 221,78,2,221,70,6,205,170,973
1100 DATA 34,221,203,7,70,40,1,235,811
1110 DATA 205,103,177,221,126,3,221,190,1246
1120 DATA 6,200,221,53,6,24,215,58,783
1130 DATA 2,176,71,166,79,120,47,166,827
1140 DATA 221,203,7,70,40,4,203,47,795
1150 DATA 24,2,203,39,245,177,119,221,1030
1160 DATA 203,7,70,40,5,35,24,1,383
1170 DATA 43,125,187,48,16,241,221,203,1076
1180 DATA 7,70,40,4,203,30,24,2,300
1190 DATA 203,22,245,24,226,78,58,3,859
1200 DATA 176,161,71,241,221,203,7,70,1150
1210 DATA 40,4,203,25,24,2,203,17,518
1220 DATA 58,3,176,47,161,176,119,201,941
1230 DATA 221,78,0,221,203,7,70,40,840
1240 DATA 5,221,70,3,24,3,221,70,617
1250 DATA 1,221,112,6,205,170,34,229,978
1260 DATA 229,221,78,2,221,70,6,205,1032
1270 DATA 170,34,209,167,237,82,125,61,1085
1280 DATA 221,119,10,221,203,7,70,40,891
1290 DATA 5,221,52,6,24,3,221,53,585
1300 DATA 6,221,78,0,221,70,6,205,807
1310 DATA 170,34,209,229,205,40,178,221,1286
1320 DATA 126,6,221,203,7,70,40,5,678
1330 DATA 221,190,1,24,3,221,190,3,853
1340 DATA 32,209,225,221,126,8,166,119,1106
1250 DATA 221,70,10,72,0,35,119,16,533
1360 DATA 252,35,221,126,9,166,119,201,1129
1370 DATA 221,126,8,205,58,178,35,19,850
1380 DATA 6,0,221,78,10,237,176,221,949
1390 DATA 126,9,71,47,166,79,235,120,853
1400 DATA 166,177,119,235,201,221,126,2,1247
1410 DATA 221,190,0,40,67,56,65,221,860
1420 DATA 126,1,221,190,3,40,57,56,694
1430 DATA 55,221,78,0,221,70,1,205,851
1440 DATA 170,34,229,221,119,4,221,78,1076
1450 DATA 2,221,70,1,205,170,34,221,924
1460 DATA 119,5,193,121,189,40,25,221,913
1470 DATA 70,4,62,0,55,31,16,252,490
1480 DATA 221,119,8,221,70,5,62,255,961
1490 DATA 167,31,16,252,221,119,9,201,1016
1500 DATA 207,25,0,0,0,0,0,0,232
```

Battre les chiffres

Notre nouveau casse-tête consiste à redistribuer par ordre croissant des chiffres en utilisant le moins de permutations possibles. Simple? Pas tant que cela.

Le programme génère de manière aléatoire une liste de nombres à trier. Il n'est possible de modifier l'ordre des nombres qu'en permutant des groupes spécifiés dans la liste. Ainsi, si l'ordinateur génère la liste aléatoire suivante en réponse à la demande du joueur de disposer de 9 nombres :

2 8 4 7 1 5 6 9 3

et si le joueur spécifie Permuter? 5, les cinq premiers nombres sont permutés, et la liste devient :

1 7 4 8 2 5 6 9 3

Cela ne devrait pas vous prendre très longtemps pour résoudre ce problème, et vous penserez qu'il peut être résolu de manière prédéfinie dans un algorithme approprié. Mais il est difficile en pratique de trouver un algorithme satisfaisant pour ce casse-tête. Supposons qu'il y ait *n* nombres dans la liste. L'algorithme le plus évident est le suivant :

- Trouver le plus grand chiffre de la liste, et renverser l'ordre des chiffres jusqu'à cette position. Le plus grand nombre est maintenant à l'extrémité gauche de la liste.
- Renverser les *n* chiffres de sorte que le nombre le plus grand soit à sa place à droite de la liste. Cela n'a pris que deux coups.
- Trouver le deuxième plus grand nombre et répéter la même démarche. Mettre ce nombre à sa place n'aura pris que « *n* - 1 » permutations.
- Répéter le même processus jusqu'à obtention de la liste ordonnée.

Cet algorithme résout *toujours* le problème en 2*n* - 3 déplacements. Il est cependant possible de trouver une solution faisant intervenir moins de permutations. Pour appréhender le gain possible en déplacements avec une stratégie d'ensemble, regardez l'exemple que nous donnons avec les cartes. Notre algorithme suppose 7 déplacements ([2 × 5] - 3), mais un joueur habile fera mieux avec 4.

Ce programme est une simple illustration d'un jeu de permutation. Vous pouvez vous essayer à développer des jeux qui permutent les extrémités d'une ligne, ou encore qui effectuent des permutations sur toute une grille et non plus seulement sur une seule ligne. Si vous créez ainsi votre propre jeu, pensez à animer les déplacements en utilisant différents blocs de couleurs pour remplacer les nombres. Vous pouvez aussi améliorer le jeu en incorporant un algorithme de repêchage d'un joueur en difficulté.

Renverser l'ordre

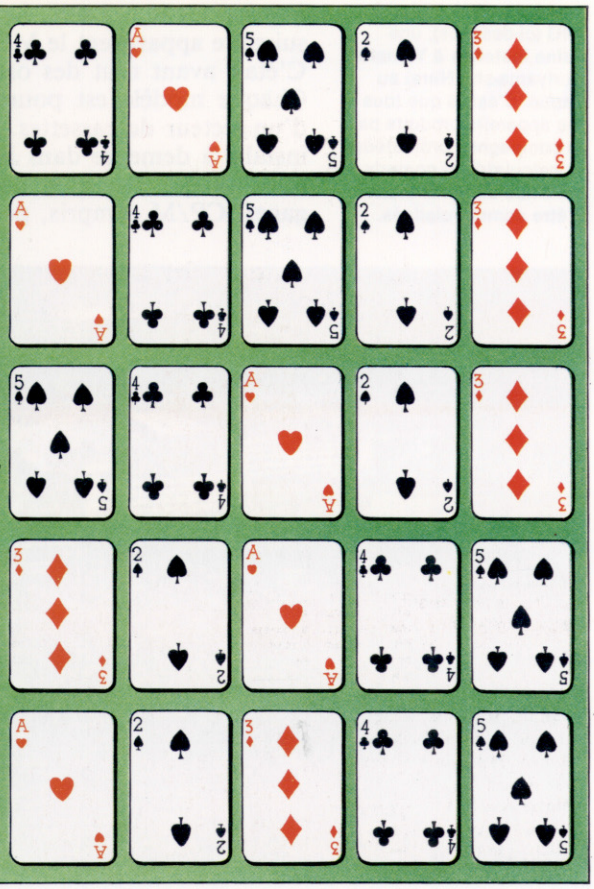
```
 20 DIM a(20)
 30 CLS : PRINT "RENVERSER L'ORDRE !"
 40 INPUT "Combien de nombres ";n
 50 IF n<0 OR n>20 OR n<>INT n THEN GO TO 30
 60 REM Mélange des nombres
 70 FOR i=1 TO n: LET a(1)=1: NEXT 1
 80 RANDOMIZE
 90 FOR i=1 TO n
100 LET r=INT (RND*n+1)
110 LET x=a(p): LET a(p)=a(1): LET a(1)=x
120 NEXT 1
130 LET d=1
135 REM Afficher l'ordre
140 CLS : PRINT "Déplacement ";d;": La liste est:": PRINT
150 FOR i=1 TO n: PRINT a(1);" ";: NEXT i
152 REM Chercher le meilleur jeu
154 LET i=1
156 IF a(i)=i THEN LET i=i+1: IF<i =n THEN GO
TO 156
158 IF i>n THEN GO TO 230
159 REM suivre une instruction go
160 PRINT : PRINT : PRINT "Permuter l'ordre?";p
170 IF p<>INT p OR p 0 OR p n THEN GO TO 140
175 REM Permuter p
180 LET d=d+1
190 FOR i=1 TO INT (p/2)
200 LET x=a(i): LET a(i)=a(p-i+1): LET a(p-i+1)=x
210 NEXT i
220 GO TO 140
230 REM Un vainqueur !
240 PRINT : PRINT :GR "Vous  avez  réussi  en  ";d;"
déplacements"
250 PRINT : INPUT "Un autre jeu (o/n) ?";a$
260 IF a$="O" OR a$="o" THEN RUN
270 CLS : STOP
```

Variantes de basic

Pour le Commodore 64 et pour le Vic-20, remplacez RANDOMIZE par XX=RND(-TI); RND*N, par RND(1)*N et CLS par PRINT CHR$(147). Pour le BBC Micro, supprimez la ligne 80 et remplacez R=INT(RND*N+1) par R=RND(N). Pour l'Oric-1 et pour l'Oric Atmos, supprimez la ligne 80 et remplacez RND*N par RND(1)*N.

Permutations

Le but de ces permutations est de trier une liste en permutant plusieurs fois de suite certains de ses sous-ensembles. La partie de la liste devant être permutée commence toujours par l'élément le plus à gauche et est indiquée par le nombre d'éléments qu'elle comporte. La bonne séquence de déplacements est ici 2-3-5-3, ce qui signifie « permutez les deux cartes les plus à gauche, puis les trois cartes les plus à gauche, etc. ».



Un géant nippon

Sharp Corporation, compagnie japonaise, propose un immense éventail de produits. En 1983, ses ventes à l'étranger ont dépassé les vingt milliards de francs...

Les origines de Sharp furent des plus modestes, même si l'histoire de cette société japonaise est constamment marquée par des innovations technologiques majeures. Elles remontent à la création, en 1915, d'une petite entreprise par Tokuji Hayakawa, désireux de commercialiser un crayon porte-mine de son invention (qu'il appela Ever Sharp, « toujours pointu », d'où le nom de la firme). Les années qui suivirent furent marquées par une expansion rapide, et, en 1925, Sharp aborda l'électronique avec un poste radio à galène. C'était ses premiers pas vers le monde de l'électronique. Mais sa véritable entrée sur les marchés internationaux de la grande consommation en matière de produits électroniques intervint après la Seconde Guerre mondiale, avec d'abord les postes de télévision et les appareils ménagers, puis, vers le milieu des années soixante, les calculatrices de bureau. Aujourd'hui, Sharp est une énorme multinationale avec trente-quatre unités de production réparties dans trente pays différents.

Le premier ordinateur Sharp apparu en Europe fut le MZ80K, lancé en 1981. L'année suivante apparurent le MZ380A et le MZ380B. C'était avant tout des ordinateurs de gestion. Chaque modèle est pourvu d'un moniteur et d'un lecteur de cassettes. Aucun langage n'est installé à demeure dans la ROM de ces appareils, ce qui veut dire que de nombreux langages, CP/M compris, peuvent y être chargés.

Industrial Instruments Group
Les projets sur lesquels travaille le centre de recherches de Sharp sont ensuite menés à bien à l'IIG (ci-dessous), une usine installée à Yamato-Koriyama-shi (Nara) au Japon. C'est là que tous les appareils produits par la compagnie, ordinateurs et calculatrices compris, sont mis au point avant d'être commercialisés.

Au début de 1983, Sharp a mis en vente une gamme complète d'ordinateurs domestiques et de gestion, avec l'introduction du PC-1500 (un ordinateur de poche) et du MZ-3541 (destiné aux milieux d'affaires). Le PC-1500, en particulier, a connu un très gros succès qui a entraîné l'apparition du PC-1251, également de poche.

Le lancement du MZ-711 a marqué l'entrée véritable de Sharp sur le marché de la micro-informatique individuelle. C'est la version européenne du MZ-700 japonais, sur laquelle toute la place occupée, en mémoire, par l'énorme alphabet japonais sur la version originale est récupérée pour accroître les capacités graphiques des modèles vendus en Europe. L'ensemble est équipé d'un enregistreur de données, et il est possible d'y raccorder une imprimante-table traçante.

Le PC-1500A est sorti en mai 1984 : c'est une version améliorée du PC-1500. Ce nouveau modèle dispose d'une RAM de 8,9 K, extensible à 24 K. Très prochainement, Sharp devrait mettre en vente le PC-1350, un ordinateur de poche à écran de quatre lignes, et possédant certaines capacités graphiques. Il y aura aussi sans doute Sharpwriter, union de la machine à écrire ZX-401 (qui servira à la fois de clavier et d'imprimante) et du micro-ordinateur MZ-3541, tous deux étant raccordés par l'intermédiaire d'une interface RS232.

Sharp compte à la fois intensifier ses efforts en direction des ordinateurs de poche, qui lui paraissent très prometteurs, et renforcer sa position auprès des amateurs de micro-informatique individuelle. Avec la famille des MZ 700, Sharp montre réellement ses intentions : s'installer durablement sur le marché des ordinateurs domestiques. Du point de vue commercial, tout cela est encore regroupé au sein d'une division chargée de l'équipement de bureau.

Sharp compte s'installer en Angleterre, précisément à Wrexham, où une usine est en cours de construction : elle produira des magnétoscopes, et devrait, en 1985, en livrer soixante mille à travers toute l'Europe.

Sharp a aussi envisagé de lancer un ordinateur au standard MSX. Mais, bien que le modèle ait fait l'objet de recherches approfondies, il n'est pas encore question de le commercialiser en Europe, du moins tant que les grandes batailles que se livrent actuellement les constructeurs n'auront pas clarifié les rapports de force.

L'ENCYCLOPÉDIE DES ARMES

LES FORCES ARMÉES DU MONDE

Pour comprendre ces terribles Armes qui font trembler le monde.

• Des documents historiques et d'actualité :
Vous saurez tout sur les combats de l'Intrépid américain... sur les Phantom au Vietnam... sur la guerre des Malouines du Tchad ou du Liban...

• Des photos, des plans techniques détaillés sur les armes militaires les plus secrètes.
Les technologies les plus avancées sur les armes offensives ou dissuasives, leur mode d'emploi, leurs performances, leur terrain d'application vous sont expliqués en détail et en maquettes.

• Des analyses d'experts sur les forces en présence dans le monde...
Vous connaîtrez la puissance d'armement de la Chine, de l'URSS, ou des USA... celle de Cuba, comme celle d'Israël, de la Lybie ou de l'Iran.

• Des croquis détaillés de chaque arme passée au crible sur toutes les faces.
Semaine après semaine, vous pourrez constituer la plus extraordinaire panoplie de documents et d'écorchés des armes de guerre du 20e siècle.

Avec plus de 6000 photos et dessins en couleurs et 200 planches techniques, semaine après semaine, l'ENCYCLOPÉDIE DES ARMES constituera une collection aussi indispensable à l'historien, au modéliste, au journaliste qu'à tous ceux qui veulent comprendre l'enjeu des combats d'hier et d'aujourd'hui.

EDITIONS ATLAS

"LE COURS DE PEINTURE"
UN CHEF-D'ŒUVRE DE SIMPLICITÉ,
UNE PALETTE D'EXEMPLES CONCRETS,
UN MAÎTRE A VOS CÔTÉS
POUR GUIDER VOTRE ŒIL,
ASSURER VOTRE MAIN,
EXERCER VOS TALENTS.

EDITIONS
ATLAS